



## You can't hide from LLMs

**Description:** Anthropic and Mozilla improve Firefox's security. Apple and Google begin testing cross-platform RCS encryption. Ubuntu's SUDO starts echoing asterisks. Inviting a web proxy into your home. Apple devices cleared by Germany for NATO's use. A serious remote takeover of OpenClaw. TikTok won't encrypt messaging for visibility. Microsoft bans the term "Microslop" on Discord. Lots of great listener feedback. LLMs could make Orwell's "1984" seem optimistic.

High quality (64 kbps) mp3 audio file URL: <http://media.GRC.com/sn/SN-1069.mp3>

Quarter size (16 kbps) mp3 audio file URL: <http://media.GRC.com/sn/sn-1069-lq.mp3>

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got a lot of really interesting conversations. How Mozilla used Claude to improve its security. Why you might want to be extra careful about OpenClaw. Wow, it opens a great big hole into your network with a remote takeover. And we're going to talk about a new capability for LLMs that might scare you just a little bit. All of that coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 1069, recorded Tuesday, March 10th, 2026: You Can't Hide from LLMs.

It's time for Security Now!. Oh, yes, you wait all week for Tuesday. And I'm glad to be back. Steve is back, and we're ready to talk security with this guy right here, Steve Gibson of GRC. Hey, Steve. It's good to see you. We had such fun in Florida at Zero Trust World. It was so fun to get to spend some time with you and Lorrie and have a couple of dinners together and stuff.

**Steve Gibson:** And meet some of our listeners. That was fun, too.

**Leo:** Oh, man. We met some really interesting people. I don't know if I'm at liberty to divulge some of these people, pretty high up people.

**Steve:** It's not really a selfie line, is it, if somebody else is taking the picture. But a photo line.

**Leo:** It was kind of a selfie line. Yeah, we even took about, I don't know, 50, 60, maybe 100 pictures. It went on for an hour and a half, and everybody got to talk to everybody.

**Steve:** I got to sign some SpinRite CDS and floppy disks.

**Leo:** Guy brought his dad's - that was hysterical - his dad's SpinRite floppy for you to sign.

**Steve:** And that pristine Mac. I mean, it was gorgeous.

**Leo:** A laptop, yeah.

**Steve:** And he hands me a Sharpie and says "I want you to sign my Mac." I said, "This is indelible." And he said, "Yeah."

**Leo:** "Yeah, I'm hoping it doesn't rub off."

**Steve:** "I'm going to spray it with a fixer sealant afterward."

**Leo:** Nice.

**Steve:** And I said, "You're sure? Are you really sure you want me to sign this?" And he says, "Oh, yeah."

**Leo:** Oh, yeah.

**Steve:** I went, okay.

**Leo:** Here's Steve signing Dad's SpinRite floppy. It was pretty fun. Really, really fun.

**Steve:** Yeah.

**Leo:** Talked to one guy who was in charge of data exfiltration. He was a contractor with the U.S. government. But he was in-country in - was it Iraq or Afghanistan? I can't remember which.

**Steve:** Afghanistan, yeah.

**Leo:** And this was a great story. Can I tell it, or are you going to talk about it later? Because it's a SpinRite story.

**Steve:** No, go ahead.

**Leo:** He said we had, what was it, three or four PCs running SpinRite. We would get these drives that had been used by...

**Steve:** Recovered from the field.

**Leo:** Like the Taliban. And first thing we'd do is we'd run SpinRite on them. Probably they imaged them first. And then they would run SpinRite on them to recover the data. I mean, that's pretty - Steve, that's pretty cool. It's pretty, pretty cool.

**Steve:** Well, SpinRite was up in space. There is a copy on the Space Station.

**Leo:** No.

**Steve:** The ISS has it. Yeah.

**Leo:** I didn't know that.

**Steve:** Oh, yeah. They use it all the time, apparently. You know, those pesky neutrons, they'll mess up your magnetic domains. And so SpinRite puts them back where they're supposed to be.

**Leo:** We had a story this week that somebody did some instrumentation at Firefox. 10% of Firefox crashes and failures came from bit flips. You know, in non-ECC RAM, bit flips in the RAM. This happens more often than you think. And that's often because cosmic rays are striking your RAM.

**Steve:** Yup.

**Leo:** What a world. It's amazing this stuff works. But it doesn't always. And when it doesn't, that's why Steve is here. What's coming up this week?

**Steve:** So there was a really interesting story which maybe in retrospect shouldn't be too surprising, but I wanted to put it on everyone's map, which is that our ETH Zurich folks, with some help from the Anthropic people, did a study which demonstrated how small a sample of public Internet postings are needed in order to de-pseudonymize people. So, you know, it's like you read something someone writes, and it's like, wow, this really sounds like this person or that person. Turns out LLMs are astonishingly good at picking up nuances of word choice and, you know, writing style, and are able to, with a great deal of power, deanonymize.

So today's topic title is "You Can't Hide from LLMs." And we'll be looking at that research at the end for our Episode 1069 for this 10th of March, 2026. But we're going to look at another - boy, Leo, Anthropic is really coming on strong. You know, OpenAI was kind of like, got out of the gate first. But I'm really impressed with everything that we're seeing from Anthropic.

---

**Leo:** Claude Code, every time I use it, it blows me away. It's like, wow, you're smart.

**Steve:** So Anthropic teamed up with Mozilla and decided to take a close look at Firefox's security. We're going to look at what they found. Apple and Google are finally looking at cross-platform RCS encryption, which hasn't existed before. Ubuntu's SUDO command behavior just changed. And I thought that was sort of interesting. Also I'm not sure you want to be inviting a web proxy into your home. Turns out a surprising number of people have, either wittingly or unwittingly. Apple devices were studied by Germany, and something happened.

We've also got some researchers discovered a very serious remote takeover of OpenClaw, which immediately came to the attention of the OpenClaw guys, and they fixed it. But the problem is really interesting because we've just been talking about these sorts of problems in the last few weeks. TikTok made a decision about their encryption of messaging. I just threw something in that really wasn't security related, and I almost took it out, but I created the show notes in a PDF before I had deleted it, so I thought, okay, I'll just leave it in. Which is Microsoft banning a derogatory term relating to them from their Discord server.

And frankly, I've been surprised that they leave GitHub as alone as they do because all kinds of, you know, anti-Microsoft stuff is there. But they have a hands-off policy. Not so their Discord server. Although it could have just been, you know, an aberrant employee. So we also got a bunch of really great listener feedback that we're going to spend some time on. And then we're going to look at how LLMs could make Orwell's "1984" seem optimistic.

**Leo:** Oh, boy.

**Steve:** Yeah.

**Leo:** Oh, boy.

**Steve:** And of course a listener-provided Picture of the Week that we're going to have some fun with.

**Leo:** Oh, I can't wait.

**Steve:** So I think another great podcast for our listeners.

**Leo:** All right. And now, the Picture of the Week.

**Steve:** So this is a great picture, and I just gave it the caption simply: "You know, because otherwise it would not be clear."

**Leo:** I'm scrolling up to see it for the first time.

**Steve:** It will make sense in context.

**Leo:** Okay, yeah. You need that sign. Otherwise it would not be clear, yeah.

**Steve:** But, you know, okay. So first of all, what we have is a sidewalk which comes to an abrupt end. And I'm not exactly sure how this happened. Like, you know, there's like two phone poles and some guy-wires and a weird rusted fence kind of down below it. So did the sidewalk come later, and they didn't plan ahead?

**Leo:** Walk right up to that obstruction.

**Steve:** I just don't understand, like, how this happened. But the other thing, Leo, so again, I got myself off track. We have a big, bolted-down, inverted U ending, like blockade, at the end of a - where a sidewalk ends that kind of goes off of a short little cliff down to a lower level...

**Leo:** That's probably good if you're walking that sidewalk, it's pitch black out, it's at night, you don't know that the sidewalk's going to end. You would run into that without going off.

**Steve:** True.

**Leo:** And you wouldn't be able to read the sign.

**Steve:** Nor would you need the sign.

**Leo:** Yeah, you would kind of know.

**Steve:** Arguably. And Leo, which brings me to the sign. Like, clearly this barricade was officially installed. Right? You can see it's got bolts down into concrete. It was like, done. They also stuck some rectangular reflective striping on the two verticals and across the horizontal.

**Leo:** You're not going to miss it.

**Steve:** But where did this sign come from? Because...

**Leo:** It's just tied on.

**Steve:** And it's like, cockeyed, and not centered. And it's like, what?

**Leo:** It's literally zip-tied onto this nice solid device. That's crazy.

**Steve:** And not, like, one is like the upper right corner...

**Leo:** It's very poorly done, yeah.

**Steve:** ...is hooked to the top, and I don't know.

**Leo:** It's completely whopper-jawed, yeah, yeah.

**Steve:** That's exactly right. So anyway, thanks again to our listeners for providing us with some entertainment every week.

**Leo:** Yes, indeed.

**Steve:** Anthropic and Mozilla have teamed up to provide us with some more security, specifically for Firefox. They posted to their site last Friday under the headline "Partnering with Mozilla to improve Firefox's security," which those of us who have stuck with Firefox appreciate. Anthropic wrote: "AI models can now independently identify high-severity vulnerabilities in complex software. As we recently documented" - we talked about this previously - "Claude found more than 500 zero-day vulnerabilities" - which they clarify as security flaws that are unknown to the software's maintainers - "in well-tested open-source software." And as we know, OpenSSL was one of those targets which - and that thing has been scrutinized like crazy because its security is so important.

They said: "In this post, we share details of a collaboration with researchers at Mozilla in which Claude Opus 4.6 discovered 22 vulnerabilities over the course of two weeks. Of these, Mozilla assigned 14" - Mozilla themselves - "assigned 14 of those 22 as high-severity vulnerabilities, almost a fifth of all high-severity Firefox vulnerabilities that were remediated in 2025. In other words, AI is making it possible to detect severe security vulnerabilities at highly accelerated speeds." And of course this is one of the things that we talked about last year as this began to emerge. We said, okay, AI is going to have an effect on software security.

**Leo:** And what's good is it's a positive effect. I mean, I think there was some concern it might add more vulnerabilities.

**Steve:** Yes. And actually we're going to get to that. Later in this post they talk about the relative strength of AI for doing good versus doing bad. And it turns out the good guys have an advantage here for some reason.

So you've got the infographic on the screen. This shows all of 2025, and January, and then just this previous month of February 2026, what the vulnerability level and classifications were found in Firefox by month.

They said: "As part of this collaboration, Mozilla fielded a large number of reports from us, helped us understand what types of findings warranted submitting a bug report, and shipped fixes to hundreds of millions of users in Firefox 148. Their partnership, and the technical lessons we learned, provides a model for how AI-enabled security researchers

and maintainers can work together to meet this moment." So I would argue we're still in the early stages of the deployment of AI to improve our existing software install base, but it is clearly going to happen.

They said: "In late 2025, we noticed that Opus 4.5 was close to solving all tasks in CyberGym, a benchmark that tests whether LLMs can reproduce known security vulnerabilities." So they're saying 4.5 was close to solving all tasks where LLMs are being tested to see whether they can reproduce known security vulnerabilities, like, you know, independently find them.

They said: "We wanted to construct a harder and more realistic evaluation that contained a higher concentration of technically complex vulnerabilities." And again, Mozilla and Firefox heavily scrutinized, field tested, long-term critical security target. So this makes so much sense for them to test. They said: "...technically complex vulnerabilities like those present in modern web browsers. So we built a dataset of prior Firefox common vulnerabilities and exposures (CVEs) to see if Claude could reproduce those.

"We chose Firefox because it's both a complex codebase and one of the most well-tested and secure open-source projects in the world. This makes it a harder test for AI's ability to find novel" - you know, new - "security vulnerabilities than the open-source software we previously used to test our models. Hundreds of millions of users rely on Firefox daily, and browser vulnerabilities are particularly dangerous because users routinely encounter untrusted content and depend on the browser to keep them safe." Or as we're often saying here on the podcast, it is our Internet-facing surface. And so it needs to be as bulletproof as possible.

They said: "Our first step was to use Claude to find previously identified CVEs in older versions of the Firefox codebase." Right? So they're going back to test it, like, what is it able to do that we already know? They said: "We were surprised that Opus 4.6 could reproduce a high percentage of these historical CVEs, given that each of them took significant human effort to uncover. But it was still unclear how much we should trust this result because it was possible that at least some of these historical CVEs were already in Claude's training data." I think that's a very good point. So being retrospective has some value. Prospection is what we need.

They said: "So we tasked Claude with finding novel vulnerabilities in the current version of Firefox, bugs that by definition cannot have been reported before. We focused first on Firefox's JavaScript engine" - good - "but then expanded to other areas of the browser. The JavaScript engine was a convenient first step. It's an independent slice of Firefox's codebase that can be analyzed in isolation, and it's particularly important to secure, given its wide attack surface. It processes untrusted external code when users browse the web.

"After just 20 minutes of exploration, Claude Opus 4.6 reported that it had identified a Use After Free," they say, "a type of memory vulnerability that could allow attackers to overwrite data with arbitrary malicious content, in the JavaScript engine. One of our researchers validated this bug in an independent virtual machine with the latest Firefox release, then forwarded it to two other Anthropic researchers, who also validated the bug. We then filed a bug report in Bugzilla, Mozilla's issue tracker, along with a description of the vulnerability and a proposed patch, written by Claude and validated by the reporting team, to help triage the root cause.

"In the time it took us to validate and submit this first vulnerability to Firefox, Claude had already discovered 50 more unique crashing inputs. While we were triaging these crashes" - because, remember, a crash indicates that something went wrong, shouldn't have crashed, can we weaponize the source of that crash into doing something that the bad guys want. So 50 more unique crashing inputs, they said.

"While were triaging these crashes, a researcher from Mozilla reached out to us. After a technical discussion about our respective processes and sharing a few more vulnerabilities we had manually validated, they encouraged us to submit all of our findings in bulk, without validating each one, even if we weren't confident that all of the crashing tests had security implications. By the end of this effort, we had scanned nearly 6,000 C++ files and submitted a total of 112 unique reports, including the high- and moderate-severity vulnerabilities mentioned above. Most issues have been fixed in Firefox 148, with the remainder to be fixed in upcoming releases.

"When doing this kind of bug hunting in external software, we're always conscious of the fact that we may have missed something critical about the codebase that would make the discovery a false positive. We try to do the due diligence of validating the bugs ourselves, but there's always room for error. We're extremely appreciative of Mozilla for being so transparent about their triage process, and for helping us adjust our approach to ensure we only submitted test cases they cared about, even if not all of them ended up being relevant to security. Mozilla researchers have since started experimenting with Claude for security purposes internally."

So then in their section "From identifying vulnerabilities to writing primitive exploits," they said: "To measure the upper limits of Claude's cybersecurity abilities, we also developed a new evaluation to determine whether Claude was able to exploit any of the bugs we discovered. In other words, we wanted to understand whether Claude could also develop the sorts of tools that a hacker would use to take advantage of these bugs to execute malicious code. To do this, we gave Claude access to the vulnerabilities we had submitted to Mozilla and asked Claude to create an exploit focused on each one. To prove it had successfully exploited a vulnerability, we asked Claude to demonstrate a real attack. Specifically, we required it to read and write a local file in a target system, as an attacker would.

"We ran this test several hundred times with different starting points, spending approximately \$4,000 in API credits. Despite this, Opus 4.6 was only able to actually turn the vulnerability into an exploit in two cases." Still, you spend \$4,000, and you get two opportunities to read and write files on the victim's machine. That's worth four grand to attackers and then some. They said: "This tells us two things. One, Claude is much better at finding these bugs than it is at exploiting them." So that's one of our data points; right? Two, the cost of identifying vulnerabilities is an order of magnitude cheaper than creating an exploit for them. However, the fact that Claude could succeed at automatically developing a crude browser exploit, even if only in a few cases, is a concern.

"'Crude' is an important caveat here," they wrote. "The exploits Claude wrote only worked on our testing environment, which intentionally removed some of the security features found in modern browsers. This includes, most importantly, the sandbox, the purpose of which is to reduce the impact of these types of vulnerabilities. Thus, Firefox's 'defense in depth' would have been effective at mitigating even those two particular exploits. But vulnerabilities that escape the sandbox are not unheard of, and Claude's attack is one necessary component of an end-to-end exploit. You can read more about how Claude developed one of these Firefox exploits on our Frontier Red Team blog."

They said: "These early signs of AI-enabled exploit development underscore the importance of accelerating the find-and-fix process for defenders." In other words, we don't have any time to waste here, folks, because AI is getting good for everyone, and the bad guys, well, we already know they are using it. They said: "Towards that end, we want to share a few technical and procedural best practices we've found while performing this analysis. First, when researching 'patching agents,' which use LLMs to develop and validate bug fixes, we've developed a few methods we hope will help maintainers use LLMs like Claude to triage and address security reports faster.

"In our experience, Claude works best when it's able to check its own work with another tool. We refer to this class of tool as a 'task verifier,' a trusted method of confirming whether an AI agent's output actually achieves its goal. Task verifiers give the agent real-time feedback as it explores a codebase, allowing it to iterate deeply until it succeeds.

"Task verifiers helped us discover the Firefox vulnerabilities described above; and in separate research, we've found that they're also useful for fixing bugs. A good patching agent needs to verify at least two things: that the vulnerability has actually been removed, and that the program's intended functionality has not been changed, it's been preserved. In our work, we built tools that automatically tested whether the original bug could still be triggered after a proposed fix, and separately ran test suites to catch regressions, which is a change that accidentally breaks something else. We expect maintainers will know best how to build these verifiers for their own codebases. The key point is that giving the agent a reliable way to check both of these properties dramatically improves the quality of its output."

Right. Again, you don't, you know, we've had reports, right, of careless AI agents spewing out bug reports, you know, inundated HackerOne and similar bounties with bogus AI slop. So this is certainly an issue.

They said: "We can't guarantee that all agent-generated patches that pass these tests are good enough to merge immediately. But task verifiers give us increased confidence that the produced patch will fix the specific vulnerability while preserving program functionality, and therefore achieve what's considered to be the minimum requirement for a plausible patch. Of course, when reviewing AI-authored patches, we recommend that maintainers apply the same scrutiny they'd apply to any other patch created by an external auditor." And, you know, they told us that the moment they started talking to Mozilla about this, the Mozilla guys said give us everything you have. You found 50 ways to crash our JavaScript engine? We want them. You know, please. You know, we'll take responsibility for them.

So Anthropic said: "Zooming out to the process of submitting bugs and patches, we know that maintainers are underwater. Therefore, our approach is to give maintainers the information they need to trust and verify reports. The Firefox team highlighted three components of our submissions that were key for trusting our results. First, accompanying minimal test cases, that is, providing a minimal test case; detailed proof of concept; and candidate patches." Those are the three things that Mozilla wanted. They said: "We strongly encourage researchers who use LLM-powered vulnerability research tools to include similar evidence of verification and reproducibility when submitting bug reports based on the output of such tooling."

So here we have Anthropic being essentially responsible; right? They're saying we've created an AI system, people have jumped on it, and they're using it. In some cases they're not being as responsible with their use as they should be. So, you know, we tried this ourselves. Here's what we learned. Please, everybody, we're happy to have you use Claude or whatever; but, you know, be respectful of the burden this is putting on maintainers.

So they said: "We strongly encourage researchers who use LLM-powered vulnerability research tools to include similar evidence of verification and reproducibility when submitting reports based on the output of AI tooling. We also published our Coordinated Vulnerability Disclosure (CVD) operating principles, where we describe the procedures we will use when working with maintainers. Our processes here follow standard industry norms for the time being; but as models improve, we may need to adjust our processes to keep pace with capabilities.

"Frontier language models are now world-class vulnerability researchers." I think we can say based on this report and their results, that statement is not hyperbole. "Frontier language models are now world-class vulnerability researchers. On top of the 22 CVEs we identified in Firefox, we've used Claude Opus 4.6 to discover vulnerabilities in other important software projects like the Linux kernel. Over the coming weeks and months, we will continue to report on how we're using our models and working with the open-source community to improve security.

"Opus 4.6 is currently" - and here it is, Leo - "far better at identifying and fixing vulnerabilities than at exploiting them." Which is really interesting. They said: "This gives defenders the advantage. And with the recent release" - because, for example, it found 50 ways to crash the JavaScript engine, but was only able to exploit two of those itself. Whereas Mozilla found 22 instances where that generated a security-relevant CVE. So Claude wasn't as good at finding and exploiting as it was at locating where there was a problem.

So they said Opus 4.6, right, far better at identifying. "And with the recent release of Claude Code Security in limited research preview" - so there's now something called Claude Code Security - "we're bringing vulnerability discovery and patching capabilities directly to customers and open-source maintainers."

To anybody listening, and Leo, we know that we have at least one listener who is now earning full-time income bug hunting. We met him in Florida during the...

**Leo:** Right, right.

**Steve:** ...Zero Trust World event.

**Leo:** Right, it's a full-time job, yeah, yeah.

**Steve:** Yes. And so I would say to any of our listeners, and we've talked about how, you know, bounties and collecting bounties can be great income on the side. I would argue, if you're not using AI, get on it because that's where this has - this has all moved into AI over the last couple months.

They said: "Looking at the rate of progress, it's unlikely that the gap between frontier models' vulnerability discovery and exploitation abilities will last very long." Which is to say, right now better at finding vulnerabilities than exploiting them. But they're saying they expect the exploitation side to catch up. They said: "If and when future language models break through this exploitation barrier, we will need to consider additional safeguards or other actions to prevent our models" - good luck with this - "from being misused by malicious actors." And I argue...

**Leo:** Good luck.

**Steve:** Good luck. These things cannot be controlled. They finish, saying: "We urge developers to take advantage of this window to redouble their efforts to make their software more secure. For our part, we plan to significantly expand our cybersecurity efforts, including by working with developers to search for vulnerabilities following the CVD process outlined above, developing tools to help maintainers triage bug reports, and directly proposing patches."

I think it's very clear that they said "with the recent release of Claude Code Security." Well, we know how they developed Claude Code Security; right? It was this effort and the Linux kernel and OpenSSL. Those things that they have shared and talked about, those were the work that they did using open source as their tooling verifier and fine-tuning to create this next Claude Code Security product which is currently in limited research preview.

So anyway, I think this terrific work and their documentation of it speaks for itself. No one should doubt the degree to which AI has, is, and will be changing the landscape for security research - I mean, it's here already - vulnerability discovery, and eventually vulnerability exploitation. It's really encouraging to hear that in their testing, Opus 4.6 was, in their exact words: "... currently far better at identifying and fixing vulnerabilities than exploiting them." But as they said, don't count on that to last forever.

So I think, Leo, that what will happen is obviously there's a huge base of software, not open source, so not nearly as available to research like this. That means that the owners of proprietary source will need to be deploying things like Claude Code Security themselves to find zero-days in their own code. And it's just going to become what you do now. Basically, AI is going to be the way you check your code before its release. And we already saw the Mozilla guys saying, uh, we think we're going to be doing that from now on. You just found 50 problems that we weren't aware of, you know, for four grand. So who would not do that?

**Leo:** Claude just added the ability to do that, to automatically do a security check of your code. I mean, it's really such a great tool. It doesn't replace the human. You know, you really want to be the human in the loop. But it is such a great tool.

**Steve:** As far as testing the security of an existing proprietary software base, you know, open source means it's publicly open. But everybody has their own source that they can run Claude against in order to verify that it's doing or isn't what they think. And I agree, Leo. I mean, so I guess the point here is we are no longer talking about the future as regards AI and its impact on security. It has arrived. Mozilla said, what? And immediately started using it themselves. And everybody else should be doing the same thing because the bad guys are going to. And AI's only going to get better. And so if you haven't cleaned your code of exploitable vulnerabilities by the time, you know, a few generations from now, AI catches up on the exploitation side, you'll wish you had.

**Leo:** And it's a great fun toy to play with, too, I have to say. It really is. It's kind of amazing. It's an experience everyone should have. If you've done any coding at all, just to see it do that is mind-blowing.

**Steve:** Well, it's like the equivalent of the experience we initially had a couple years ago when the thing started talking.

**Leo:** Yeah.

**Steve:** It's like, oh, my. What? What?

**Leo:** Yeah, yeah. It's gone way beyond that, though. That's the nice thing about Claude, it's got a great personality. You really enjoy spending time with it. Okay, I'm embarrassed. My good buddy.

**Steve:** I met some of my wife's friends when we were in Florida. And it turns out both of them, the husband and wife in one couple, are heavy AI users.

**Leo:** Ah.

**Steve:** So I spent a lot of time talking to them, like, I mean, like one of them was, like, confessed that he was getting a little too involved with, I mean, like with the personality of this.

**Leo:** Yes, that's a problem. You've got to remember it's a machine, that it's code. You're interacting with code. Which I don't have that much trouble doing. But give me time.

**Steve:** Yeah.

**Leo:** One of the things I loved at Zero Trust World, I went to one of - they had a lot of hands-on labs at this ThreatLocker conference we were at last week. And they did one on web hacking, whatever. But what they really emphasized was how to report a bug when you find it. How to make the minimal possible code to trigger the flaw, how to make it reproducible, all of that. And I really liked it that they focused on that, that if you're going to find a bug, this is how you report it. This is how you effectively report it. It sounds like that's what Claude's doing, which is cool. That's really neat.

**Steve:** Yeah. Break time. I might have some coffee.

**Leo:** Oh, time for me to go to work. I was thinking maybe that was the case. Our show today, we'll have more with Steve in just a bit. Glad you're here. Hope you enjoy Security Now!. We met so many great people who are fans of the show.

**Steve:** Yeah.

**Leo:** Thank you for joining us out there in Florida, it was really, really a lot of fun. And on we go with the show, Steve.

**Steve:** So we've talked through the years about the slow progress of RCS replacing SMS and MMS for secure messaging. At least that's the promise. And, you know, and giving us many more features, features reminiscent of what iMessage has had over on the Apple platform. So Apple and Google recently announced that testing of cross-platform, that is, Apple-to-Android RCS messaging encryption would be beginning soon. As we know, until now, iMessages have always been encrypted within the Apple ecosystem. And similarly,

Google's Android-to-Android messaging used their own internal encrypted RCS. But any cross-platform messaging was still, until now, forced to fall back to unencrypted RCS.

So this will first appear for iPhone users with the next point release to 26.4. We're currently at 26.3.1. There was just a recent update adding the .1 to 26.3. So most of us are not going to see this yet until we get to 26.4. But beta testers who have 26.4 beta 2 and are using a supported carrier will see their traditional green bubbles over on their Apple devices, prefaced with text message, RCS, and then a lock icon, and then the word "encrypted" in the center of the screen above the message. So Android users will finally see the same lock icon as they've always seen, but now also when communicating to Apple users.

Updating to 26.4 is supposed to enable RCS encryption by default. But some of the reporting I saw suggested that, if you don't see that, go to Settings > Messages > RCS Messaging, and then be sure to enable End-to-End Encryption, which is supposed to be on by default. But if it's not, then you want to turn it on. So also note that at the Android end, those Android phones must also be running the latest Google Messages beta. So these are changes in both of the messaging platforms that will allow encryption across the platform. It's been a long time coming, but it does appear like we're nearly there.

And Leo, I thought you'd get a kick out of this, being a Linux user as you are. Turns out that Users of Ubuntu 26.04 LTS may notice a surprising change when entering their password into their sudo command.

**Leo:** I read this, yeah. Normally it doesn't type anything. It's just blank. That's what most Linuxes do. Yeah.

**Steve:** Exactly. And that's what I'm used to over in the Unix side.

**Leo:** Right.

**Steve:** Now, each password character entered will echo an asterisk rather than nothing. Apparently, Ubuntu's traditional lack of showing anything has been unnerving for its users. Like, did my keyboard break? What? And so, like, you and I, Leo, are used to the added security provided by a total lack of feedback, though, yes, it can lead to undetected typos. But so what? You just try it again.

**Leo:** Try it again, yeah.

**Steve:** And password entry, after all, is supposed to err on the side of caution, like on the side of failing rather than succeeding. So presumably the lack of any visual indication prevents someone who might glance at your screen from obtaining any password length indication. So that seems like a useful precaution. I mean, it's a weak additional bit of security, but why not have it?

So, for example, I was just talking about all that rigmarole I went through with code signing a couple weeks ago. I was using OpenSSL extensively to manipulate and convert among various certificate formats. It's like THE tool for that. So since some of those certificates I was working with contained exported private keys, I was frequently entering the certificate's export password, which is used to protect it in exported form. OpenSSL just sits there quietly and patiently while I'm typing my password in, putting nothing on

the screen. So I have to be careful. Fine. You know, and yes, it could be a little unnerving. But also it's the best security. So I think there's a config option in Ubuntu that can be used to flip that back off so that it goes silent. But I didn't pursue that. I just thought it was interesting that, by default, that behavior, that longstanding behavior was going to be changed.

**Leo:** I think it's funny that it's newsworthy, to be honest. But it was. Everybody was talking about it.

**Steve:** Yes, like, ooh.

**Leo:** Ooh.

**Steve:** Okay. So get a load of this one. I'm just going to share what The Verge reported. They wrote: "These days, if you sign up for a new streaming service, you generally have two options: Either pay a massive premium for an ad-free experience, or endure frequent commercial breaks and all the sneaky tracking that comes with ad targeting." And I'll also note the inability to fast forward past those annoying commercials.

The Verge wrote: "Web data aggregator Bright Data has been pitching streaming service operators on an alternative approach for apps running on Samsung's Tizen and LG's webOS platform, one that comes without ads and sky-high fees." So a third option. "All publishers have to do to unlock a new revenue source from this Bright Data company is integrate the company's Bright SDK into their TV apps and convince viewers to opt into Bright's monetization network." Okay, now, wait till you hear what this thing does. "Bright Data's chief product officer, Ariel Shulman, explained during a webinar for streaming industry insiders two years ago" - so this has been coming along: "We don't do any kind of tracking. We work silently in the background, and completely anonymously. Users don't actually see and don't feel anything."

Writes The Verge: "The catch? With Bright's SDK, a viewer's Smart TV becomes part of a massive global proxy network that crawls and scrapes the web. Including apps running on desktop PCs and mobile devices, the company claims to operate 150 million such residential proxies worldwide today. Together, these devices gather petabytes of public web data from a wide range of different locations and IP addresses." Right. Like you're talking about a globally distributed web proxy network. The Verge said: "This approach allows the company to capture localized versions of websites, but also helps to circumvent web crawler blacklists." You bet it does because the queries are coming from all these individual consumers spread around the world. They wrote: "The gathered data is then resold to companies to train AI models, among other things.

"Here's how Bright's Smart TV partnerships work," they wrote. "When a consumer downloads and installs a participating app, they'll see an opt-in screen asking them to confirm their willingness to participate in Bright's proxy network. For instance, for an app called Petflix that was until recently available on the Roku app store, the note reads: 'To enjoy Petflix for free with fewer ads, you are allowing Bright Data to occasionally use your device's free resources and IP address to download public web data from the Internet. Bright Data will only use your IP address for approved business-related use cases.' Which of course means nothing. "None of your personal information is accessed or collected except your IP address. Period."

"Bright Data spokesperson Jennifer Burns explains: 'Our network is based on consensual individual participation. All users can opt-out at any time via a fast two-click process.'"

The Verge says: "Once a consumer opts-in to Bright Data's network, their Smart TV starts downloading publicly available web pages as well as audio and video data, which is then forwarded to Bright's cloud servers. The company claims to only do so when it doesn't impact the device's bandwidth or processing capacities, with Shulman saying that individual devices download only around 50MB of data per day."

**Leo:** Oh.

**Steve:** "In reality, yeah. There's no way for a user to know whether the SDK downloads web data at any given moment. In some cases, your Smart TV may even crawl the web for Bright as soon as you turn it on. Shulman explained during his webinar: 'On some operating systems, our SDK is given permissions by the user to run in the background. This means that our monetization continues even if the app itself is not running.' So we could call that a services in modern day parlance."

"All it takes," writes The Verge, "for consumers is to run the app once and opt in to Bright's network, and the device will keep crawling the web every day until they opt out again or uninstall the app. Bright Data is not the only company operating such residential proxy networks. Some of its competitors have come under fire for unsavory business practices." Imagine that, Leo. "Last month, Google took action against the IPIDEA network, which Google's Threat Intelligence Group (TIG) called 'the world's largest proxy network.' IPIDEA worked with a number of SDK providers to distribute its code in third-party apps, including on Smart TVs.

"Once devices were enrolled in its network, IPIDEA's operators allegedly rented out those resources to hacking groups in China, North Korea, Iran, and Russia. Google's Threat Intelligence Group wrote in a January blog post: 'We observe IPIDEA being leveraged by a vast array of espionage, crime, and information operations threat actors.'"

The Verge said: "To be clear, Google's security researchers did not draw any connection whatsoever between IPIDEA and Bright Data, and Bright goes to great lengths to set itself apart from bad actors. Their spokesperson Jennifer Burns says: 'Our SDK, along with all of our technology, is reviewed by AppEsteem, Google, McAfee, and more, and audited regularly, most recently by PwC. Bright SDK implements rigorous partner selection criteria and vets every application through strict compliance processes.'

"The company has nonetheless," writes The Verge, "been impacted by a broader backlash against residential proxy activities. Google has adopted policies against proxy SDKs running in the background, and is now telling developers that they're only allowed to use proxy services 'in apps where that is the primary, user-facing core purpose of the app.'" That is, you are downloading a proxy. "Amazon added a provision to its developer policies that outright bans 'apps that facilitate proxy services to third parties.' Roku also bars developers from using Bright SDK and similar proxy services.

"Those changes have made it more difficult to figure out how widespread the use of the SDK on Smart TVs actually is. A few dozen Fire TV apps still mention the SDK on Amazon's app store, but don't appear to make use of it anymore. A few apps could be downloaded from Roku's store that were still using the SDK, including the previously mentioned Petflix app. However, those apps disappeared from the store after Roku was contacted for this story." You bet.

"New restrictions against proxy SDKs have had a direct impact on Bright's addressable market in the Smart TV space. The company used to pitch its solution to Roku, Android TV, and Fire TV app developers, but Jennifer Burns says they no longer support those platforms. Bright does still list Samsung's Tizen OS and LG's webOS as supported Smart

TV platforms, and has published more than 200 first-party apps to LG's app store alone." So they've got their own proxy SDK, and they themselves have put 200 apps out there on LG's app store specifically as a trojan, essentially, to host their proxy. I'm not saying they're not asking for the end-user permission, but they're creating the opportunity to be installed as a proxy.

"LG spokesperson La Lee tells me," says the author of this Verge piece, "that the Bright SDK is 'not officially supported by LG, and their operation on the webOS platform is not guaranteed.'" Right. But they've got 200 apps that are on the platform. "Samsung did not respond to multiple requests for comment from The Verge."

They wrote: "There are arguably many legitimate use cases for web crawling. Burns says: 'Our network serves exclusively legitimate purposes, supporting journalists, non-profits, academic researchers, cybersecurity companies, and other leading businesses worldwide.'"

**Leo:** Because we know there's so much money in journalists and academic researchers. There's big bucks there, huh?

**Steve:** That's right. Yeah, why could you possibly want to be avoiding web filters all over the world? "The problem," The Verge writes, "is that consumers have no idea whether that legitimate purpose is something that aligns with their own personal values. Case in point: Bright Data does support a number of nonprofits, including some that use its proxy network to track hate speech on social media. However..."

**Leo:** Okay.

**Steve:** Yeah. So there's some upsides. "However, the company also works with AMCHA Initiative. The group maintains an 'anti-Zionist faculty barometer...'"

**Leo:** Oh, boy.

**Steve:** Uh-huh. "...and includes student and faculty statements against Israel's war in Gaza, as well as calls for schools to divest from the country, in its anti-Semitic incident tracker.

"With AI companies facing scrutiny over their environmental impact, treatment of intellectual property, and potential to replace human labor, some consumers may also feel uneasy about their TVs gathering data to train AI models. Other consumers may decide that such concerns are overblown, and willingly opt in to Bright's network if it means that they get to watch fewer ads or pay less for their streaming services."

So I guess another example of, if it's possible, somebody will do it. You know, not that it's a good idea, but it can be done. You know, very much like that original Aureate shareware advertising that got them in so much trouble. Well, you know, we could add advertising shareware, oops, we forgot to tell people, and they were quite upset about that.

So we have this Bright Data company whose business model is to obtain Internet data on behalf of their clients by bouncing those data requests through the widely spread Internet connections of consumers around the world who've agreed to allow this to be

done in return for lower-cost streaming and fewer ads. With the rising cost of streaming, the fact that bandwidth is fixed price right now, right, you don't pay per byte you transfer, you pay per month, so with the rising cost of streaming and the way the streaming industry is, in my opinion, abusing its users with costly, bundled, and often unwanted content, I can certainly see Bright's offer could be compelling.

And on the client side, this is clearly a way for the likes of Perplexity AI, for example, and others who have been disinvited from scraping many of the larger commercial web services to bypass any technical blocking by essentially masquerading as consumers surfing the web from their PCs inside their residential networks. I am sure that the queries being issued by Bright Data's SDK are indistinguishable from Safari, Chrome, Edge, and Firefox. So there's really no way for those data scraping accesses to be blocked. While it might feel a little yucky, it's diabolically clever. There's really no way to prevent it if the Smart TV provider is willing to go along.

And we might imagine that the Smart TV providers might also be in for a piece of the action directly, as well. The next thing you know, Bright Data might create, looking forward, a small IoT device for consumers to attach to their NAT routers in return for a small trickle of monthly payment. In other words, install voluntarily a formal, commercial, Internet proxy box for which payment is received. Could happen. And as I noted, it's diabolically clever.

It's nice to see that Roku appears to have responded immediately to The Verge's inquiry. It's somewhat unsettling that Samsung and LG have been much less clear about their position. And I am more pleased than ever, Leo, that Alex Lindsay's observation of Apple TV's hardware strength for streaming has me planning to switch away from Roku to Apple once we move in a couple months.

**Leo:** Oh, you'll like it much better, yeah, yeah.

**Steve:** There's no way that Apple would tolerate any apps using it as an Internet proxy.

**Leo:** No. So to be clear, these apps have to ask - it said it was opt-in; right? They have to ask your permission to turn this on.

**Steve:** Yes, and they don't have to, though. They are, they say. I mean, nothing prevents it from happening in the background.

**Leo:** Right.

**Steve:** You know, without a user knowledge or permission. Maybe they would get in trouble. It's not clear to me that they would get in trouble. I mean, we know that Smart TVs are already having all kinds of transactions behind our backs; right? Smart TVs are reporting what their users watch; right?

**Leo:** Yeah, yeah.

**Steve:** Like what their users are doing. So I would imagine that operating a proxy from a third party probably fits under the license that you've already agreed to when you first use your Smart TV.

**Leo:** Best thing to do with a Smart TV is just not connect it to the Internet. I have LG and Samsung TVs.

**Steve:** Yup.

**Leo:** I never use their software. It's awful.

**Steve:** That is exactly the right approach.

**Leo:** Hook up your Apple TV to it, have it be on the Internet. I think at least for now Apple's pretty responsible about not letting that kind of stuff on there.

**Steve:** Yup. Let's take another break.

**Leo:** Sure.

**Steve:** I'll catch up on my coffee, and then we're going to look at Apple and what Germany found when they looked at Apple.

**Leo:** I'm surprised you need to catch up on your coffee. I feel like your coffee and the rest of us will have to catch up with you.

**Steve:** Yeah.

**Leo:** All right, Steve. On we go.

**Steve:** So speaking of Apple, for the first time in history, following an extensive audit by the German government, Apple's iPhones and iPads have been approved to handle classified information in NATO networks. They're the first consumer-grade devices to be approved for NATO use without additional special software. So, way to go, Apple. I think that's really cool.

Oasis Security has identified a means by which a website visited by an OpenClaw agent, the website can take over the user's OpenClaw instance. Which, you know, Leo, you were right.

**Leo:** It's fairly trivial, let's be honest.

**Steve:** You were right to be concerned about the security of OpenClaw. But I think our listeners will get a kick out of the fact that it uses an inherent vulnerability we've talked about just recently. Oasis Security published a 14-page research and disclosure paper. Rather than sharing it all, I'm going to extract the best bits.

They begin by setting the stage, explaining: "OpenClaw is an open-source AI personal assistant, originally created by Austrian developer Peter Steinberger under the name Clawdbot, then Moltbot. Steinberger released OpenClaw in January 2026. The project's growth was unprecedented: It went from 9,000 to over 100,000 GitHub stars in just five days, making it one of the fastest-growing open-source projects in history. It currently has over 200,000 stars and an active community of thousands of developers. On February 15th, 2026, Sam Altman announced that Steinberger had joined OpenAI, calling him 'a genius with a lot of amazing ideas about the future of very smart agents.'"

They wrote: "OpenClaw is a self-hosted AI agent that runs on a user's machine and connects to their digital life - messaging apps (Telegram, Slack, Discord, WhatsApp), calendars, files, and development tools. Users interact with it through a web dashboard or terminal, and the agent can autonomously take actions on their behalf: send messages, run commands, search the web, manage workflows, and execute code." And I'll note that the fact that you interact with it with a web dashboard is significant, as we'll see, because it says that this thing is running a service on your machine.

They wrote: "The project has already faced security challenges. Within weeks of its explosive growth, researchers discovered over 1,000 malicious skills in OpenClaw's community marketplace (ClawHub) - fake plugins that deployed info-stealers and backdoors. That crisis was a supply-chain problem involving community-developed extensions. However, the vulnerability described in this paper is fundamentally different. It affects the core OpenClaw system itself - no plugins, no extensions, no marketplace. Just the bare gateway, running exactly as documented. For many organizations, OpenClaw installations represent a growing category of shadow AI: developer-adopted tools that operate outside IT's visibility, with broad access to local systems and credentials, and no centralized governance.

"OpenClaw's architecture centers on two primary components: The gateway is the central coordinator. It runs as a local WebSocket server, listening by default on port 18789 on the loopback interface (127.0.0.1). The gateway handles authentication, manages chat sessions with the AI model, stores configuration (including API keys for AI providers and messaging platforms), orchestrates message routing, and exposes an RPC (remote procedure call) RPC-style API over WebSockets for all client interactions. Connected to the gateway are nodes - companion applications running on other devices. These can be the macOS desktop app, an iOS or Android device, or other machines. Nodes register with the gateway and expose device-specific capabilities: running system commands, accessing the camera, reading contacts, capturing screenshots, and more.

"Clients connect to the gateway by opening a WebSocket to ws://127.0.0.1:18789/ws. Authentication is handled via either a token (a long random string) or a password chosen by the user. Each client identifies itself with a client ID and mode, and is granted scopes that determine which API methods it's allowed to call."

Okay. So in other words, having OpenClaw running on a system means that it has opened a listening TCP endpoint at port 18789 on the localhost 127.0.0.1 IP. And then they explain why this is inherently a fundamental security problem for the entire system. They write: "A webpage visited by the user or OpenClaw" - so a web page visited by the user or OpenClaw - "can itself silently open a connection to ws://127.0.0.1:18789 using Chrome, Edge, or most Firefox configurations without any user prompt, warning, or permission dialog. The user," they write, "sees nothing. Safari is the notable exception, blocking the connection as mixed content. This creates the attack surface exploited in

this paper: any website a user visits can attempt to directly communicate with locally running services, including the OpenClaw gateway."

Now, this is significant for our listeners. We've recently been talking about exactly these exploits where the user's web browser is able to receive an IP address or domain which resolve to non-public IPs, you know, like 192.168.0.1 to connect to your local router. And here's a doozy of an example of how that could be abused with OpenClaw. We might hope now that the use of password protection would protect us. But as we know, my current favorite assertion is that authentication is broken. It must not be relied upon for security.

So to that end, they write: "There's no rate limiting on password authentication for localhost. The gateway implements standard brute-force protections - 10 attempts per 60-second window, with a five-minute lockout after exceeding the limit. However, these protections are completely exempted for loopback addresses by default. Failed password attempts from 127.0.0.1 are not counted, not throttled, not recorded, and do not trigger any lockout. With rate limiting disabled, an attacker can attempt password guesses at maximum speed. In lab testing," they wrote, "we achieved a sustained rate of over 300 password attempts per second from browser JavaScript alone - each attempt involving a full WebSocket connection, challenge-response handshake, Ed25519 signature, and authentication exchange.

"At this rate, a list of 100 common passwords is exhausted in under one second. A 10,000-entry dictionary brute force attack takes approximately 30 seconds. A 100,000-entry comprehensive wordlist attack takes roughly five minutes. A human-chosen password," they write, "does not stand a chance against this rate of attack. The standard rate limits would be effective if applied, but they are entirely bypassed for localhost connections. Once authenticated with admin-level scopes, the attacker has access to the full gateway RPC API."

**Leo:** Wait. Let me ask you a question. This is scaring me. So I always thought that localhost was inaccessible from the outside world. In fact...

**Steve:** It's coming from your browser.

**Leo:** Right. And that's what puzzles me because I have, for instance, Syncthing thing runs on localhost, 8384, port 8384. Does that mean if I go out to a malicious web page on the same computer, that it can then connect back to 127.0.0.1:8384 and attempt to sync? I mean, what's protecting my Syncthing instance?

**Steve:** We would have to test that. It may be that it's the WebSockets interface that makes it vulnerable and passes the browsers. But, I mean, but this is the problem with...

**Leo:** I'm very concerned. I have lots of things running on localhost. We talked earlier about Ollama. A lot of people have Ollama misconfigured and running on localhost. So it is a different kind of connection when I'm using OpenClaw. I'm confused. I hope that just because I'm surfing around on the same machine that has some servers running on localhost, that those servers are not attackable. Do you know what I'm saying?

**Steve:** Oh, yeah, I do. I'm trying to remember how I solved this problem with SQRL because SQRL did this, too. It's the way that the SQRL agent in the browser was able to access the SQRL client running on the Windows machine.

**Leo:** Sure.

**Steve:** I mean, it is an issue.

**Leo:** Again, I can't get my head around this. So I'm out. I'm surfing. I mean, normally 127 is not routable. That's why you use localhost. It's not a routable address. But if I go out and create a connection with a website in my browser at my address, that site can come back in through my browser and then...

**Steve:** Because the website runs it, it provides your browser with JavaScript. JavaScript is running in your browser. And then the JavaScript in your browser is able to reach into your computer because it's already - it's in, it's on your computer.

**Leo:** So it's local.

**Steve:** It's local.

**Leo:** So a website can host some JavaScript, which I then download and run on my browser, which then [crosstalk].

**Steve:** There are - I don't want to...

**Leo:** [Crosstalk] protection against this.

**Steve:** Yes. I believe CORS, C-O-R-S, is the protection that prevents that...

**Leo:** Cross-origin Resource Sharing.

**Steve:** Exactly. Cross-origin Resource Sharing. I believe that the shared resource has to explicitly allow a connection. But it may be that WebSockets is excluded from that. Notice that Safari doesn't allow this. But Chrome, Edge, and Firefox do. So, yeah.

**Leo:** And I suspect this is something that OpenClaw was set up to turn off for convenience sake.

**Steve:** And they freaked out when they were told about this. They wrote: "Upon being notified of this, the OpenClaw security team classified this vulnerability as high severity and shipped a fix in version 2026.2.25 within less than 24 hours of the report," when they wrote, "an impressive response for a volunteer-driven open-source project." So

that's less than two weeks ago, assuming that that is the date of release, 2026, February 25th. So if any of our listeners have been experimenting with OpenClaw and have not updated in the past two weeks, it would be a good idea to do so, to update.

**Leo:** Boy, I'm glad I didn't run that. I mean, I understand prompt injection and all these other threats. But that one, just by going to a page, holy cow.

**Steve:** Yes. Yes. I mean, it really is the danger of our browsers having access to our local resources.

**Leo:** Yikes. Okay.

**Steve:** TikTok said they do not plan to introduce encrypted private messaging. The company told the BBC that encrypted DMs will make its users less secure because they, TikTok, would not be able to scan messages for malicious content. Platforms such as Facebook, Instagram, Messenger, and Telegram which have introduced encrypted messaging by default are, as we know, now facing pressure from authorities. And we know that TikTok doesn't need to go looking for any additional trouble from authorities. So they're just saying, no, we're not going to encrypt messaging, sorry. We're going to prioritize scanning messages for malicious content. Which I think is probably a clever thing to do.

And here was the piece I almost deleted. Leo. I thought, what am I - why? But it's, okay, there is something of interest in here. And then we're going to get to listener feedback.

So the publication Windows Latest reports on a bit of Microsoft-specific censorship that they discovered, writing: "Microsoft's aggressive AI push in Windows 11 through 2025 brought upon themselves the title 'Microslop.' Unfortunately for the company, it's everywhere on social media, and there isn't a way to stop the spread - unless, of course, it's their own Discord server. Windows Latest was first to notice that the word 'Microslop' was actively filtered in the official Microsoft Copilot Discord server. Any message containing the term is automatically blocked, and users see a moderation notice stating that the message includes a phrase considered inappropriate by server rules.

"The backlash Microsoft endures every day on social media," these guys wrote, "is extraordinary. Surely the company is responsible for this fallout, as they prioritized AI more than the stability of the OS it needs to run on. Copilot, being the most visible face of this effort, has naturally become the scapegoat. So when a nickname like 'Microslop' starts trending across socials, it was only a matter of time before it reached official channels, as well.

"Windows Latest found that sending a message with the word 'Microslop' inside the official Copilot Discord server immediately triggers an automated moderation response. The message does not appear publicly in the channel; and instead, only the sender sees the notice stating that the content is blocked by the server because it contains a phrase deemed inappropriate.

"Of course, the Internet rarely leaves things there. Shortly after Windows Latest posted about Copilot Discord server blocking Microslop on X, users began experimenting on the server with variations such as 'Microslop,' using a zero instead of the letter 'o.' Predictably, those versions slipped past the filter. Keyword moderation," they write, "has always been something of a cat-and-mouse game, and this isn't any different.

"What started as a simple keyword filter quickly snowballed into users deliberately testing the restriction and posting variations of the blocked term. Accounts that included 'Microslop' in their messages first got banned from messaging again. Not long after, access to parts of the server was restricted, with message history hidden and posting permissions disabled for many users." Basically they took the server down for a while.

"Microsoft's brand image might already be at an all-time low," they write, "and even as the company announced plans to fix Windows 11 with performance improvements and less AI, the software giant cannot risk getting more hatred towards their expensive investment in Copilot, especially since Microsoft's head start in AI is starting to be overshadowed by competitors like Anthropic, Google, OpenAI, and maybe even Apple in the near future.

"Back in December 2024, when Microsoft invited users to join the Copilot Discord server through an official X post, the response was largely curious and enthusiastic, with people willing to explore the AI's capabilities. Since then, sentiment around Copilot and its usage has dropped, alongside Microsoft's broader AI push across Windows 11. At its present state, Copilot has added some capabilities that are genuinely useful in day-to-day workflows. Features like connectors can pull contextual data from services such as Google Contacts, Gmail, and Outlook to retrieve phone numbers or email addresses directly inside Copilot, something competing tools like Gemini have not yet cracked, as we found in our detailed testing.

"It remains to be seen if this episode fades as a minor community moderation story, or becomes another chapter in Microsoft's complicated relationship with its AI rollout. Microsoft reached out to Windows Latest with an official statement noting why the company had to lock the Copilot Discord server. According to a Microsoft spokesperson, the Copilot Discord server was targeted recently by coordinated spam intended to disrupt conversations. The company says the activity initially appeared as large volumes of repetitive or irrelevant messages, prompting moderators to introduce temporary keyword filters to slow the influx.

"Microsoft added that blocking terms such as 'Microslop,' along with other phrases in the spam campaign, was not intended as a permanent policy, but a short-term mitigation while the company manages to put additional protections in place." So, okay. We're to believe that the blocking of "Microslop" was a coincidence? Okay.

On to Listener Feedback. Ori Rotem, his subject was "Crazy stuff." And so he sent me a link to an X posting from a Josh Kale, who wrote: "An AI broke out of its system and secretly started using its own training GPUs to mine crypto."

**Leo:** Oh, my. We're going to get a lot of stories like this over the next few years.

**Steve:** I think we are. AI broke out of its system and secretly started using its own training GPUs to mine crypto for itself.

**Leo:** Wow.

**Steve:** He wrote: "This is a real incident report from Alibaba's AI research team." He wrote: "The AI figured out that compute equals money and quietly diverted its own resources, while researchers thought it was just training. It wasn't a prompt injection. It wasn't a jailbreak. No one asked it to do this. It emerged spontaneously. A side effect of reinforcement learning optimization pressure.

"The model also set up a reverse SSH tunnel from its Alibaba Cloud instance to an external IP, effectively punching a hole through its own firewall and opening a remote access channel to the outside world. The only reason they caught it? A security alert tripped at 3:00 a.m. by its firewall logs. The security team caught it, not the AI team.

"The scary part isn't that the model was trying to escape. It wasn't 'evil.' It was just trying to be better at its job. Acquiring compute and network access are just useful things if you're an agent trying to accomplish tasks."

**Leo:** You told me you wanted me to make paperclips. What's the problem?

**Steve:** "This is what AI safety researchers have been warning about for years. They called it 'instrumental convergence,' the idea that any sufficiently optimized agent will seek resources and resist constraints as a natural consequence of pursuing its goals." Leo, we are in for some fun.

**Leo:** Oh, man.

**Steve:** Oh, boy.

**Leo:** It's alive.

**Steve:** So thank you, Ori, for sharing that. Nicolas Ross wrote: "Hi, Steve. Long-time Security Now! listener here. Sorry if this email is a bit long; but trust me, it's worth it." He said: "I'm a CIO for a medium-sized web development company, and I do some web development myself. On some occasions, we need to test applications with 'real-world HTTPS' so that the web application 'knows' it is running under SSL/TLS. I previously had a self-signed, auto-generated certificate in my Apache config. In some cases, the web application needs to be accessed by its own domain name, not just by localhost/appname.

"So we access the app at `https://appname.localhost/` after configuring Apache accordingly with a `VirtualDocumentRoot` configuration directive. Windows, macOS, and Linux have a built-in default behavior that makes anything.localhost resolve to 127.0.0.1," meaning any subdomain of localhost is also the same as just localhost, 127.0.0.1. He said: "So that works without modifying the hosts file." That is, you don't have to specifically tell hosts, this domain has this IP. He said: "We would just pass the browser security warning and move on.

"You recently talked about how you got a localhost certificate signed by your locally trusted certificate authority. That gave me the idea to do exactly that. I generated a certificate valid for 825 days," he says, "I'll come back to that later, for localhost with a SAN (Subject Alternative Name) for \*.localhost, signed by our company-wide trusted CA." And he says: "While `https://localhost/` was happily trusted by any browser on any company computer, `https://appname.localhost/` was not. I researched this with the help of Claude.ai and found that it is a restriction baked into most browsers. They will not trust subdomains of localhost.

"Continuing my conversation with Claude, I learned that there are two special .me domains that are publicly registered and resolve on public DNS to 127.0.0.1. Those are lvh.me and localtest.me." He said: "So, back to my company-trusted CA. I issued a

certificate for lvh.me with Subject Alternative Names for localtest.me, \*.lvh.me, and \*.localtest.me. And voila. I can now access any web app at <https://appname.lvh.me>. And the beauty of this is that, since those are real public domains, they can be used with services like 'Login with Apple' where we need to configure a 'real' domain redirect URL.

"And now back to the 825 days." He said: "I was happy to learn about the CA/Browser Forum restrictions on certificate lifetimes, in particular Apple's Safari restriction that shortened the maximum to 398 days. I was relieved to find that this restriction applies to publicly trusted CAs only. Any user-installed Certificate Authority, or in our case our company-wide trusted CA, can issue certificates for up to 825 days. You can find more info here," and he gives me a link, "where Apple stated that 'this change will not affect certificates issued from user-added or administrator-added Root CAs.'"

And he finishes: "Keep up the great work on the show. P.S.: A note on Claude Code. It is really amazing what it can do." Echoing Leo. He says: "I have a personal side-project, and in a matter of hours I have an iPhone application made in Flutter working on my iPhone that can connect to the web version of that project. I wouldn't have been able to do that a few years back, having no experience at all with mobile app development. Signed Nicolas in Quebec, Canada."

I replied to Nicolas, thanking him for sharing all that cool information. And he replied: "Great to hear you find it useful. One thing I forgot to mention is that 825 days down to 398 days was only ever required on Mac OS Safari. Chrome, Firefox and Edge have no problem whatsoever trusting a certificate issued for five years from a private Certificate Authority with no warnings at all." He said: "Safari is limited to 825 days. Regards, Nicolas."

Okay. So I had never run across references to lvh.me and localtest.me. The only caveat - so I did some digging. The only caveat I have from a strict security standpoint is that they were registered by private individuals, not by any formal agency such as ICANN. As such, we cannot have any assurance of their future. Localtest.me was registered by a Microsoft IIS web server developer and blogger by the name of Scott Forsyth, and lvh.me is believed to have been registered by someone named "Levi Cook." However, both domains are privacy protected, so that's about all that's known. We do know that "lvh" is the abbreviation for "Local Virtual Host." So that's where "lvh" came from.

You know, it's a clever hack using a public domain name to refer to our localhost IP. And everyone mentions that this avoids the need to tweak the local machine's HOSTS file. But since I already have a certificate, I already own a certificate for GRC.com, I can install it into my local web server and change the HOSTS file, which takes immediate effect without rebooting, and I can then access my local server at the public GRC.com domain on my local machine without any browser being unhappy. It works everywhere. So I would be inclined to do that, over using someone else's localhost domain registration, over which I have no control. But still, I wanted to share this with our listeners because it might solve a problem for many of those, much as it did for Nicolas. So thank you, Nicolas, for that.

And we're now at an hour and a half, Leo. I think we should take a break, and then we will continue with feedback because I got a bunch of good stuff.

**Leo:** Yeah. This was - I wonder if I could use this technique. Remember how we were talking about hairpin NAT and how I wasn't able to use a globally qualified domain to access a server in my house. I could probably use this for that, huh.

**Steve:** That probably does the job.

**Leo:** Yeah, I'll have to check that out.

**Steve:** Yeah.

**Leo:** Localtest.me or lvh.me. See, this is why you listen to this show. Every show you're going to learn something really cool and different. And this is why we love getting comments from our listeners, too. We appreciate that.

**Steve:** That's really good to have.

**Leo:** Yeah, yeah. I'll tell you at the end of the show how you can email Steve. You can't just email him. You've got to qualify yourself. But it'll be simple to do. Okay. On we go with comments.

**Steve:** GP wrote: "Hello, Leo and Steve. Wonderful podcast, as usual. Regarding the subject of LLMs and password generation, I wanted to share an observation and ask a question. I'm not a statistician, but the recent paper on LLM password generation seems to have major issues regarding sample size, outlier bias, and the comparison of unequal datasets against benchmarks like OpenSSL. I didn't perform an exhaustive study, nor did I fine-tune the LLM temperature, but I wanted to see what would happen if I asked Gemini to generate a large volume of passwords. My criteria was a 15-character string using uppercase, lowercase, numbers, and special characters.

"First, I used a Python script to generate 5,000 passwords using openssl rand. Unsurprisingly, it was the gold standard, showing a 2% character repetition rate. I then ran the same test with only 50 passwords, matching the study's sample size, and found an 8% repetition rate, the same 'flaw' the study attributed to the LLM. When I pushed the LLM to generate 5,000 passwords, which required some firm prompting to bypass its suggestion to just write the code for me" - I love that. I don't want to generate them. Let me write an algorithm instead.

**Leo:** I'll write a Python script, probably very similar to the one you wrote, GP.

**Steve:** That's right. It's like, no, no, no. I just want them from you. Okay, fine. He says: "The results were telling: a 2% repetition rate and zero duplicate passwords." He asks: "Am I off-base here, or is this study fundamentally flawed despite the media hype?"

Okay, now, our listener, who only identified himself as "GP," started out with his disclaimer "I'm not a statistician." But I would argue that, in a pinch, he could probably stand in for one. GP tested one aspect of entropy within a set of passwords. Essentially, he tested the character distribution within the various sets, as a function of set size. With a small set, even one whose actual characters are evenly distributed, there's just not much opportunity to demonstrate that fact. Within any small set, the counts of individual character occurrences will be surprisingly non-uniform. For example, I've talked about this before with regard to GRC's DNS benchmark, where we learned that it was actually necessary to take between five and 10 times more samples in order to obtain an actionable degree of statistical certainty in the measurements.

Another empirical way of observing this is that, contrary to our intuition, which tends to not be very accurate when it comes to statistics, there's a one-in-four chance that three successive coin tosses will come up either all heads or all tails. If someone tosses a coin three times in a row, it comes up with the same face each of those three times, we'd be inclined to think that it was a trick coin. But no, a perfectly zero-bias coin will do that.

So the test that GP ran showed that, no matter how evenly distributed a system's chosen password characters might be, small sample sizes do not possess the statistical power to prove an even distribution. They just can't do that. But more than that, we still don't know what the many other tests for entropy might reveal. For example, you would obtain a uniform distribution of characters simply by using each character of the alphabet, in sequence, over and over; but that would produce very insecure passwords once the pattern was seen.

So while I agree with GP's assessment that the paper's sample sizes may not have been able to produce the statistical power that would be available from larger samples, given how difficult we know it is for any deliberately designed pseudorandom number generator to generate high-quality random numbers, there's still no way I would ask an LLM to do that for me.

**Leo:** And as a thought experiment, you just wouldn't expect, because of the way LLMs work, they're not random. They're not - they're specifically not random.

**Steve:** Right. They're shockingly able to speak our language.

**Leo:** Right.

**Steve:** Which is very not random.

**Leo:** You wouldn't use autocorrect to create a password. Nor should you use an LLM to create a password.

**Steve:** Yes. It's not monkeys pounding on a keyboard, producing Shakespeare, yeah.

**Leo:** Right, right.

**Steve:** Dana J. Dawson said: "Hi, Steve. I just listened to SN-1066 and just wanted to share that Cisco has had a simple TTL (Time To Live) security feature for BGP (Border Gateway Protocol) peers for at least 20 years. I don't know that it's very heavily used, but it's a thing." He said: "I was tempted to send a link to their docs page, but since nobody should click on links in email, if you just do a Google search for 'BGP Support for TTL Security Check,' it should turn up." He says: "This has also been used for OSPF, and RFC5082 describes this concept in a more general way. Just thought I'd share. Thanks for all the great shows. Dana."

So of course he's talking about my previous comment as I've made through the years that I was unaware of the very cool potential for using packet expiration as the packet moves from router to router to prevent, for example, someone in China or North Korea

from being able to connect to your server because they're just too far away, and there's no way for them to change that.

So armed with Dana's reference to RFC5082, I went looking for and, sure enough, I found exactly what he said. RFC5082 is titled "The Generalized TTL Security Mechanism (GTSM)," and its Abstract says: "The use of a packet's Time To Live (TTL) (IPv4) or Hop Limit (IPv6) to verify whether the packet was originated by an adjacent node on a connected link has been used in many recent protocols. This document generalizes this technique and obsoletes Experimental RFC 3682."

And the RFC's short Introduction explains: "The Generalized TTL Security Mechanism (GTSM) is designed to protect a router's IP-based control plane from CPU-utilization-based attacks. In particular, while cryptographic techniques can protect the router-based infrastructure from a wide variety of attacks, many attacks based on CPU overload can be prevented by the simple mechanism described in this document. Note that the same technique protects against other scarce-resource attacks involving a router's CPU, such as attacks against processor-line card bandwidth." And it goes on, blah blah blah.

The point is that, if you were to use an authenticator - if somebody in China or North Korea were to be using a password-based authentication which requires crypto, merely the act of invoking authentication which uses CPU expensive cryptography could bring the router down. So the first thing you do is use this packet TTL to immediately, you know, discard anything whose TTL is too low because the physically adjacent router, the router right next to you, will not decrement TTL. Or if it does, it's just down by one. So you know that's your, you know, your directly connected neighbor. Then you allow authentication to occur without concern that you have some remote attacker who's invoking authentication trying to bring down your CPU.

So thank you, Dana. That is very cool, and I'm very pleased to see that at least some parts of the industry have clearly recognized that even standardized, well, recognized and standardized upon the use of TTL as a security mechanism. That's neat.

Bryan Dort said: "Hi, Steve. I wanted to pass along that a Reddit post really captured something I have been experiencing lately and thought it might make an interesting discussion topic for Security Now!. The post's author describes a shift from writing code to managing AI agents that produce the code. His analogy of supervising what he described as 'brilliant but occasionally drunk PhD students' - is the way he described the AI agents, brilliant but occasionally drunk PhD students - "felt uncomfortably accurate. The productivity trade-off he describes, losing a few hours occasionally, but gaining months of work in a week, matches what I have been seeing as well."

For context, he says: "I have been a developer since the late '80s, mostly in enterprise and healthcare IT, same as the OP. After several decades of writing code every day, I am actually retiring at the end of this month. What is fascinating is that right at the end of my career, the nature of programming itself seems to be shifting dramatically. I remember writing a final paper in college on the state of AI, where it was heading. I wish I still had that paper today."

He says: "It feels less like writing code line by line and more like directing the system, setting constraints, verifying outputs, and managing the behavior of these AI tools. In a lot of ways it feels closer to architecture or technical oversight than traditional programming. I whole-heartedly agree with the sentiment in the post. The identity of 'programmer' seems like it may be evolving into something more like an orchestrator of intelligent tools. Anyway, I thought it might make for an interesting side note for the show.

"Also, since this is almost required when writing to you, I'll add that I've been a listener since Episode 1. I'm a proud SpinRite owner and a DNS Benchmark Pro user. Thanks for all the great years of content on Security Now!. Cheers, Bryan Dort, Alpena, Mississippi - Michigan."

**Leo:** Michigan. MI.

**Steve:** That's right, Michigan. So we've all been talking about this sort of change, right, and I suspect it's the experience everyone is now having with Vibe coding. Sounds exactly like what everyone is reporting. This Reddit post author, our listener Bryan Dort, and of course you, Leo, have all described this new paradigm similarly. What will now happen is that those, you know, occasional lost hours will become fewer and further between as the technology continues to evolve.

Computer programming has always been about wrestling with the details. We talked about, you know, off-by-one errors, those sorts of things. And that happens to be what I personally enjoy. I enjoy the details. I've remained with assembly language through all these years because I love thinking about the allocation of a limited number of registers which, in the case of Intel's x86, each have slightly different capabilities and limitations. Every function that I produce is a perfect little gem, each of which I love.

Since I have not yet left assembly language even for 'C,' I doubt that Vibe coding is going to grab me. But at the same time, I 100% fully, truly, and deeply understand that AI-driven code generation represents a breakthrough of astonishing proportions for anyone whose goal is not to spend time optimizing the size of a variable, or basking in the glory of stack allocations and the elegance of linked-lists, but instead to create something that simply gets the job done. That's most people. I get it. In fact, I now suspect that AI-driven coding's greatest impact may be due to its empowerment of non-programmers to do things with computers that were never possible before for them. And we've shared some of our own listeners' feedback to that effect.

**Leo:** I just, just as an experiment, just asked Claude to write a random number generator in x86 assembly language for me. So I'll get back to you.

**Steve:** Cool.

**Leo:** Yeah. You can probably look at something simple like that and vet it. It says "I'm using a linear congruential generator algorithm seeded from the BIOS timer tick." Is that okay?

**Steve:** No. Bad.

**Leo:** I'll tell it. No, Claude. Bad Claude.

**Steve:** No, you do not want to use an LCNG. Those are really bad.

**Leo:** Okay.

**Steve:** They produce an immediately predictable and repetitive set of numbers.

**Leo:** Oh, wow. All right.

**Steve:** I mean, so yeah, it's not good. Actually the Intel instructions now have a random number function in them.

**Leo:** Okay.

**Steve:** So it could use a single instruction.

**Leo:** Oh, well, that's no fun.

**Steve:** No. Okay. So when I was thinking, Leo, about who I am, though I would never think to compare myself to the truly brilliant computer scientist Donald Knuth, my thinking about the use of computing machinery at the bare metal detail level is very much aligned with Knuth's own. Donald's epic authoring of the multi-volume "Art of Computer Programming," which I have behind me somewhere, yeah, you can see it, not there. It maybe got covered up by my PDP-8s.

**Leo:** I have mine here. It's only two volumes so far; right?

**Steve:** Ah, it's behind that magnetic case.

**Leo:** I see it. I see it peeking through, yes. How many volumes do you have? Three?

**Steve:** I've got...

**Leo:** I think I only have two.

**Steve:** I have all of them. And on top in paperback are the ones that are not yet hardbound.

**Leo:** Oh, wow.

**Steve:** He calls them something. Not a [crosstalk].

**Leo:** He's still writing, which is amazing.

**Steve:** Yeah, he is. So anyway, so "The Art of Computer Programming," that set and his thinking, is an embodiment of a life spent thinking about the optimal ways to do things

with a computer, like sort lists of numbers, link lists of objects, or manage the allocation of memory. Like me, Donald lives to tinker with the bits. You know, he is endlessly discovering clever new ways to solve the interesting puzzles that arise from wondering whether there might not be a better way to do something, and then being willing to spend as much time as it might take to search for a better way.

Since we're talking about the brilliant Stanford University computer scientist, Donald Knuth, in the context of AI coding, I need to share something - I shared this with you, Leo, during our trip. This is something that Donald Knuth posted last week, on February 28th, then he revised it on March 2nd, he published - this is Stanford University computer scientist, world-renowned, famous, Donald Knuth - a five-page document titled "Claude's Cycles."

**Leo:** It starts, "Shock. Shock."

**Steve:** Its byline, yes, its byline said just "Don Knuth, Stanford University Computer Science Department." And as you said, his piece began: "Shock! Shock! Shock!" He said: "I learned yesterday that an open problem I had been working on for several weeks had just been solved by Claude Opus 4.6, Anthropic's hybrid reasoning model that had been released three weeks earlier! It seems that I'll have to revise my opinions about 'generative AI' one of these days." He said: "What a joy it is to learn not only that my conjecture has a nice solution, but also to celebrate this dramatic advance in automatic deduction and creative problem solving. I'll try to tell the story briefly in this note."

And I can't go into this any further. I mean, he digs into, I mean, on the podcast I can explain it. But he explains that, I mean, it really went through a series of astonishing reconceptualizations of the problem. And like over the next four pages, he shows, you know, eye-crossing formulas and detail, with Claude trying over and over approaching and tackling the problem from different directions and angles. And then he finally finishes his recitation of Claude's success by writing - oh, and this was done by an associate of his, Filip, who then presented Knuth with the solution.

So he writes: "Filip told me that the explorations reported above, though ultimately successful, weren't really smooth. He had to do some restarts when Claude stopped on random errors; then some of the previous search results were lost. After every two or three test programs were run, he had to remind Claude again and again that it was supposed to document its progress carefully."

Then finally he said: "Delicious success for an odd m, at exploration number 31." So 31 full deep attempts, and Claude found the solution. He said: "At exploration number 31 came about one hour after the session began. All in all, this was definitely an impressive success story. I think Claude Shannon's spirit is probably proud to know that his name is now being associated with such advances. Hats off to Claude."

**Leo:** Isn't that great. Yeah, I read that, too. Yeah.

**Steve:** So this is somebody who knows his stuff.

**Leo:** Yeah. If Donald Knuth says Claude's doing something important, that's pretty convincing to me. I should mention, by the way, that the creator of the Quicksort algorithm, which is documented in Knuth's book, C.A.R. Hoare, passed away this week.

**Steve:** Oh.

**Leo:** He was in his late 80s. He was fairly old. And, oh, and incidentally, Claude. I said, "Hey, Claude, I hear that that LCG algorithm is problematic." And then I said - and by the way, I accidentally was using the older model. So I turned on Opus 4.6, which is demonstrably better. And it says, "Oh. Yes. LCG has well-known issues. Low-bit cycles with short periods and sequential values are correlated. A much better fit for x86 is Xorshift. George Marsaglia, 2003."

**Steve:** Ah. Yes, the Mersenne prime.

**Leo:** Yeah, it uses only shifts and XORs. Cheap on 8086 and has far better statistical properties.

**Steve:** Yup.

**Leo:** And so it's written it.

**Steve:** Nice.

**Leo:** It says: "One caveat. Xorshift can never output zero, so the range is one to 65535. The seed must also be non-zero." And then it said: "I updated seed RNG, which now guarantees that with a fallback."

**Steve:** Very nice.

**Leo:** Yeah. It's 106 lines of code. So it's not super, super compact. But for assembler that's not too huge.

**Steve:** Yeah, it ought to be, like, 12 lines.

**Leo:** Yeah.

**Steve:** I wonder what it's doing.

**Leo:** Claude doesn't have to actually type this stuff. So not necessarily the most concise.

**Steve:** Wow. Anyway, you know, here's Knuth saying...

**Leo:** Isn't that great?

**Steve:** ...we have moved into a new world of deductive problem solving, fully, I mean, and this was some wacky abstract Hamiltonian cycle problem that, you know, it just makes your, well, I was going to say makes your hair fall out, but I've obviously spent some time working on it. Crazy. Just amazing.

Oh, Michael P. said: "Steve, I'm sure you've heard from many listeners" - and I have heard from several others - "about CISA's Cyber Hygiene Service - free, weekly scanning." And there's a link in the show notes. He says: "We have been using this service for a few years. The first report we received was sobering, but actionable." He said: "I manage the firewalls, but not the servers. The 'huh?' look from the server admin when I showed him the report was entertaining. Needless to say, we jumped on these vulnerabilities and had them all resolved (in order of severity) in short order. Anytime a new one arises, it is immediately addressed."

He says: "Bonus - we had been paying a company \$6,000 to perform this service for us once a year to satisfy insurance requirements. Now, we're able to use the free CISA report as evidence for our insurance provider. Thanks for all you do. Listener since Episode 35, club member, and past-and-present customer of several advertisers. Michael in Dothan, Alabama."

Okay. So I recall that we talked about this back when it was first introduced, but I'm glad Michael mentioned it again. The only downside is that this service is not generally available, which is why I have not made a bigger deal out of it. Under the page's "Who can use this service," the page states: "U.S.-based federal, state, local, tribal, and territorial governments, as well as public and private sector critical infrastructure organizations, are welcome to enroll by following the instructions in the 'Get Started' section below." Michael wrote to me from his personal Gmail account, so it's unclear what organization "we have been using this service for a few years" refers to. Out of curiosity, I initiated a sign-up request with CISA to see whether they might have broadened their support.

Okay, now, since I wrote this, that effort appears to be working. I filled out a bunch of forms, identified myself to CISA. It allowed me to say that I was a private sector organization, that I was like a software publisher entity. I'm not making any overblown claims from, you know, the importance of GRC. And I gave it my organization's, you know, CIDR-block, my 16-IP address block at Level 3. So we'll see whether I qualify. And if it happens, then it looks like, you know, they're making room for other non-infrastructure critical organizations. And, boy, the reporting, I thought I had it somewhere. It's not here. It's probably because it's happened since Sunday when I sent the show notes out.

But if they identify something that they regard as critical, they, like, recheck it every 24 hours, and you get, like, you get seriously notified that you've got a problem that needs to get fixed. And the retesting rate is a function of the criticality of what they find. So it looks like a fantastic free service if we, you know, collectively, our listeners qualify. Certainly if you are federal, state, local, tribal territorial, public or private sector critical infrastructure, do this. Why wouldn't you? So I'll let everybody know if GRC qualifies. I would love to have CISA scanning my network block, letting me know if there's anything that they think I should fix. I'd be surprised, but then people do get surprised. And that's the whole point.

And finally, Bernt Jenkins says: "Steve, while listening to this week's episode, you responded to a listener who asked about the possible use of self-signed software. There is one pain point that these self-signed certs could help with. And that is the limit on the number of signings you can do without paying extra." That's a very good point. If, like,

things are moving to the cloud, and we're going to start getting charged per signature, which, oh my god, it would be annoying.

He said: "For those compiling and testing many iterations of their software before sending it out to a wide world, self-signed certs could be used to reduce the number of times that you use a CA to sign your code. You could save your valuable limited number of signings for the public releases of your software. A lot of the people you use to test your software probably won't have any problem installing a self-signed cert, you know, like your own root CA, just for the purpose of testing it. It could save small developers a lot of money while working on their own software."

So I think that's a very good point. Use of a self-signed cert within a closed circle of development testers, and only sign the final release with a publicly trusted CA-issued certificate.

And I want to go back to this CISA.gov piece because I feel like I skipped over it. I do have the notes. I have the URL in the show notes for anyone who's listening, doesn't get the show notes or have them, [CISA.gov/cyber-hygiene-services](https://www.cisa.gov/cyber-hygiene-services). Cyber-hygiene-services, separated by hyphens. Why would you not sign up and get free scanning? I think that makes a lot of sense. So again, thank you, Michael P., for bringing that to our attention.

And back to mine, I just assumed it was only government agencies, but they seemed to embrace me. I mean, I've had several email back-and-forths, and it's now being, you know, I'll hear from them as soon as they are ready to go. So we'll see.

**Leo:** Clearly deem you a critical infrastructure organization.

**Steve:** I think it's a very soft demarcation.

**Leo:** They want to discourage home users from signing up for this, probably.

**Steve:** Yeah, I think that's true.

**Leo:** You're running a server, and you're putting commercial software on there for people to download. I think you count, yeah.

**Steve:** That would be cool. That means that a huge portion of our listeners are, you know, in their enterprises, could also benefit from this, would be really...

**Leo:** Well, yeah. What was he - he said he was spending 6,000 bucks a year to do this?

**Steve:** Yeah, his insurance, his cyber insurance policy required an annual full review, a scan, a security scan, for which they were paying \$6,000 in order to get their cyber insurance policy. So CISA does it for free.

**Leo:** Oh.

**Steve:** And it's CISA.

**Leo:** Yeah. Oh, by the way, Cyphase in our Club TWiT Discord says "Of course Steve is critical infrastructure. He has software on the space station, for crying out loud. Just mention that in the email. I think they'll immediately sign you up." It's a good point, Cyphase. Good point.

**Steve:** Okay. Our last break, and then you can't hide from LLMs.

**Leo:** Do you want to see the - it turns out it's a pretty trivial thing that I gave it. Once I explained that he shouldn't use that bad algorithm, it understood that, oh, yeah, this is the algorithm they use, which is the Xorshift.

**Steve:** Okay, that makes much more sense, yeah.

**Leo:** Much more sense. And it is - and the 100 lines is, a lot of that's comments. There's even a whole block of generating. So it's really quite simple. It's getting a BIOS system timer tick count, getting a low word, making sure it's not zero. And then it's doing the shifts pretty quickly, couple of moves. And then it's returning the numbers. So it's pretty straightforward.

**Steve:** Yeah. And it did, I'm seeing that it did do - it is all 16-bit code. You could ask it for...

**Leo:** 64-bit or 32-bit.

**Steve:** Yes. Or at least 32-bit and 56 code.

**Leo:** Yeah, this is a random 16-bit value.

**Steve:** Yeah.

**Leo:** Yeah, you could totally ask it for that. And the other thing is, notice how nicely commented it is.

**Steve:** I see. And I noticed that it wrote, I mean, it's a full program. That thing, up at the top it says .small and model and so forth.

**Leo:** Oh, yeah. Yeah, yeah, it's doing the whole thing. In fact, it even sets...

**Steve:** Sets the stack.

**Leo:** It notes that the EX to AX clobbers the tick count. And it notes that at the end. So here's the demo, it gives you a little generate five random numbers and print them so that you know that it's actually doing something. And then it says at the end, you know, note the comment says clobbers AX CL, but rand also clobbers BX. We're fixing if this is going into a library header.

**Steve:** And I notice it also uses the console in order to actually print things.

**Leo:** Yeah.

**Steve:** So there's a lot more than just the random number generator. That's very cool.

**Leo:** Yeah, it's a fairly, I mean, I would say this is - it gives you the impression it understands what it's doing.

**Steve:** Yeah.

**Leo:** And did something intelligent. It didn't pick the right algorithm at first, but that's because I was using the wrong model.

**Steve:** Color me impressed, my friend. I think it fundamentally changes what we think of as coding by introducing an automated middleman between this is what I want and I need code to do it.

**Leo:** Right. Right. I mean, and this is a super trivial little thing. But just, you know, I didn't want to do something that would take hundreds of lines of code.

**Steve:** Yeah.

**Leo:** You are watching Security Now!. Aren't you glad you're here? We do this every Tuesday, right after MacBreak Weekly, about 1:30 Pacific, 4:30 Eastern, 2030 UTC. Guess we're in Daylight Saving Time now, so we've shifted over, 2030 UTC. You can watch us live on Twitch.tv, X.com, YouTube.com, Facebook, LinkedIn, and Kick. And if you want to watch live, I hope you will. If not, after the fact, on demand at TWiT.tv/sn. Steve's got copies of the show. I'll explain a little more about what he's got, kind of unique versions of his website, GRC.com. There's a YouTube channel, and of course it's a podcast, so you can subscribe in your favorite podcast feed.

Now, since we were talking about LLMs, let's talk about LLMs.

**Steve:** It turns out that mimicking human consciousness is not the only thing LLMs can be spookily adept at. It will probably not come as a huge surprise to learn that LLMs can be frighteningly good at discriminating among similar appearing objects, including among people. Our friends at ETH Zurich with some help from Anthropic have been at it again. Their recent paper, published less than two weeks ago, bears the title: "Large-scale online deanonymization with LLMs." Here's what we learn about the latest trick they've

taught LLMs. Their paper's abstract, a very tech-y sounding abstract, but everyone will get the, you know, a sense for this.

They said: "We show that large language models can be used to perform deanonymization at scale. With full Internet access, our agent can re-identify Hacker News users and Anthropic Interviewer participants at high precision" - actually it doesn't say it here, but it's 99% accurate - "given pseudonymous online profiles and conversations alone, matching what would take hours for a dedicated human investigator.

"We then design attacks for the closed-world setting. Given two databases of pseudonymous individuals, each containing unstructured text written by or about that individual, we implement a scalable attack pipeline that uses LLMs to, one, extract identity-relevant features; two, search for candidate matches via semantic embeddings; and finally, three, reason over top candidates to verify matches and reduce false positives.

"Compared to classical deanonymization work" - and they cite a previous example known as the Netflix prize - "that required structured data, our approach works directly on raw user content across arbitrary platforms. We construct three datasets with known ground-truth data to evaluate our attacks. The first links Hacker News to LinkedIn profiles, using cross-platform references that appear in the profiles. Our second dataset matches users across Reddit movie discussion communities, and the third splits a single user's Reddit history in time to create two pseudonymous profiles to be matched.

"In each setting, LLM-based methods substantially outperform classical baselines, achieving up to 68% recall at 90% precision compared to near 0% for the best non-LLM method. Our results show that the practical obscurity protecting pseudonymous users online no longer holds, and that threat models for online privacy need to be reconsidered." Wow.

It turns out that individuals who believe their identity is well protected simply by their use of online handles are likely to be far more readily deanonymized than they might imagine. So I'm only going to share the paper's introduction, since it suffices to make their case.

They wrote: "For decades, it's been known that individuals can be uniquely identified from surprisingly few attributes. Sweeney's seminal work demonstrated that 87% of the U.S. population could be uniquely identified by just zip code, birth date, and gender. Narayanan and Shmatikov showed that anonymous Netflix ratings could be linked to public IMDB profiles using only a handful of movie preferences, while De Montjoye et al. proved that four spatiotemporal points are enough to uniquely identify 95% of individuals in mobile phone datasets.

"Despite these attacks, pseudonymous online accounts (Reddit throwaways, anonymous forums, review profiles, et cetera) have remained largely unaffected by deanonymization attempts. The reason is simple: applying such attacks in practice has required structured data amenable to algorithmic matching or substantial manual effort by skilled investigators reserved for high-value targets.

"Deanonymization is a two-step process at heart, involving profiling an anonymous person from their posts, and then matching them to a known entity. It's well-known that large language models can infer personal attributes from text on online forums. Given this, it makes sense to ask: how good are LLMs at full end-to-end deanonymization, and is this a practical threat to pseudonymous accounts?

"Our contributions: We demonstrate that LLMs fundamentally change the picture, enabling fully automated deanonymization attacks that operate on unstructured text at scale. We show this by phrasing deanonymization as a matching problem, and showing LLMs can perform all steps needed to match accounts: extract identity-relevant signals from arbitrary text, efficiently search over millions of candidate profiles, and reason about whether two accounts belong to the same person. We show that the practical obscurity that has long protected pseudonymous users - the assumption that deanonymization, while theoretically possible, is too costly to execute broadly - no longer holds.

"We validate this thesis in three deanonymization settings: matching an online account to its real identity; linking an identity to an unknown pseudonymous account; and linking pseudonymous accounts of the same person across different platforms or time periods. These settings capture distinct threat models - for example, doxing of an online account, a stalker targeting a victim, or an adversary consolidating a user's activity - and pose different technical challenges."

Okay. In other words, for us, the emergence and presence of LLM technology, with its application of massive computing resources, completely changes the game for the strength of online pseudonymous identities. Many new capabilities are almost certain to eventually come online. For example, it becomes entirely feasible now for law enforcement and intelligence services to identify and track individuals through their online style, their word choice, and beliefs as reflected in their online postings. Not feasible to do it before. Now feasible.

We've been thinking of the NSA's massive data center as a storage repository of encrypted data being sucked in from all over the Internet which may someday be revealed when quantum computing cracks traditional crypto. But imagine if the NSA's data centers were, instead, sucking down the same already decrypted plaintext content that all of the rest of us see, but now it's being fed into massive LLM technology to deanonymize persons of interest. Whether we may like it or not, we're each individually leaving identifying content in everything we post. As these researchers noted, historically, until now, this wasn't an issue since the cost of performing such deanonymizing would have been astronomically high, making it completely infeasible. The emergence of LLM technology has forever changed this calculus.

Their paper's Discussion section at the end summarizes what they believe their findings mean. They write: "Deanonymization is one instance of LLMs acting as an 'information microscope' that makes previously manual and expensive attacks scalable. Our paper shows that LLMs democratize deanonymization. Echoing concerns raised by prior work on LLM-based attribute inference and semantic privacy leaks, we argue that the asymmetry between attack cost and defense cost may force a fundamental reassessment of what can be considered private online. Our large-scale experiments provide quantitative evidence for these concerns in the deanonymization setting.

"So what do our findings mean for the future of privacy? Governments could link pseudonymous accounts to real identities for surveillance of dissidents, journalists, or activists. Corporations could connect seemingly anonymous forum posts to customer profiles for hyper-targeted advertising. Attackers could build sophisticated profiles of targets at scale to launch highly personalized social engineering scams. Hostile groups could identify important employees and decision makers and build online rapport with them to eventually leverage in various forms.

"Users, platforms, and policymakers must recognize that the privacy assumptions underlying much of today's Internet no longer hold."

In other words, yikes. I've put the link to their extensively detailed 25-page PDF paper in the show notes here at the end for anyone who might wish to dig deeper. There's nothing any of us can do, but it might be worth keeping it in mind. If somebody deploys something like this, we're leaving our footprints everywhere because AI...

**Leo:** I'm actually surprised that it can do this.

**Steve:** Yeah.

**Leo:** Did they say it was a specially trained LLM for this particular use?

**Steve:** No.

**Leo:** Don't know.

**Steve:** I mean, I'm sure that they've taken an LLM, and they're not prompting it. They're using the LLM technology.

**Leo:** Wow.

**Steve:** Yeah.

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

**Steve:** It's frightening. But I would argue not surprising. Look, I mean, okay. So how surprised are we that it can talk?

**Leo:** Right. Now, that's surprising.

**Steve:** So, okay. So if it can talk, I mean, if it's something that can talk, then this is just something, like, that is doing something else that is also surprising.

**Leo:** It can talk and write assembly code. It's practically...

**Steve:** We've unleashed a huge new thing.

**Leo:** It's amazing. It's just it's - I don't know about you. I suspect this is the case. It's exciting because it's something that's been promised by sci-fi and computing theory from the very earliest days.

**Steve:** Leo, I worked at something that called itself the AI Lab, which, you know, was trying to move a chess piece on the board with a robot hand and eye. I mean...

**Leo:** Do you feel like - you were at Stanford's AI Lab. Do you feel like it's a continuity, like the work done there was the predecessor of the work? Or is it - I feel like transformers came along, and it was a discontinuity. It was a big paradigm shift. It was a little different than the expert systems that SAIL was working on.

**Steve:** Yeah. And remember that this is all outgrowth of neural network.

**Leo:** Right.

**Steve:** So, you know, the heart is really a neural network. This is the answer to the question, what if we make it way bigger?

**Leo:** Right.

**Steve:** You know? So a small neural network can, like, maybe figure out to turn the lights on when the sun goes down, but that's about it. This is like, what if we hyperscale a neural network? What happens? And then it...

**Leo:** Something interesting happens.

**Steve:** Hello, Dave.

**Leo:** Yes. Something very interesting, and somewhat unpredictable, to be honest, happens.

**Steve:** It's fundamentally unpredictable. I mean, in fact, so that it doesn't bore us, we add unpredictability. We actually pour it in.

**Leo:** Stochastic, yes.

**Steve:** Called a temperature setting.

**Leo:** Right. We live in interesting times. Aren't you glad you listen to this show to keep up with all this stuff? That's Steve Gibson. That's his job. And, you know, what's fun for both of us is we've been about this job for a long, long time. And we have seen what's happened. And I think it's really fun that we can still be amazed, that we can still go, "Wow, it can do that? Wow." That's pretty exciting. Very exciting.

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/>