



Fast16.sys

Description: Bitwarden's CLI hit with a supply-chain attack. Commercial routers in Iran fail shortly before the war. Meta logging all employee activity to train replacement AI. GRC's DNS Benchmark Release 5. Two miscellaneous AI thoughts. A bunch of terrific listener feedback. Unraveling the diabolical history of "Fast16.sys."

High quality (64 kbps) mp3 audio file URL: <http://media.GRC.com/sn/SN-1076.mp3>

Quarter size (16 kbps) mp3 audio file URL: <http://media.GRC.com/sn/sn-1076-lq.mp3>

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with a great Picture of the Week and an amazing spy story, 21-year-old malware that only now has been discovered. The story of Fast16.sys coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 1076, recorded Tuesday, April 28th, 2026: "Fast16.sys."

It's time for Security Now!, ladies and gentlemen. Get ready, here's the hero, the star of our show, Mr. Steve Gibson. Hello, Steve.

Steve Gibson: Oh, Leo.

Leo: Oh, no.

Steve: We have a story today.

Leo: You know, the title is a little weird.

Steve: Yes. It is. The title of today's podcast is "Fast16.sys."

Leo: That sounds like Fat16.sys.

Steve: Yeah, not fat. Maybe it was meant to look like it.

Leo: I think so, so you would scan over it and not...

Steve: Or maybe a quick - it could have been like a math library alternate. But it's kind of a believable name.

Leo: Right.

Steve: It turns out that, through a weird, just coincidental discovery, some security researchers uncovered evidence and then proof of a diabolical, I mean, diabolically brilliant thing that predated Stuxnet by about five years.

Leo: Oh, that's why it's .sys, because that's the old Microsoft designation [crosstalk].

Steve: Oh, it's still, still. So, yes, the system32 directory is full of .sys files.

Leo: Dot sys files, huh.

Steve: So what they found was - okay. Our listeners have heard me worry that maybe the U.S. is not up to speed because we keep, you know, having reports about being attacked by China and North Korea and Russia and so forth. And I'm wondering can we give as much as we get. I'm not that worried anymore.

So today we're going to talk about, once we get to it, this amazing evidence trail and what was discovered in place back in the Windows 2000/XP days. And oh, Leo, you are going to love what this thing did because, I mean, it's just - it's so clever and so just wonderful.

Anyway. We're going to talk about Bitwarden's command line interface hit by a supply chain attack and why no Bitwarden users, you know, end users were impacted by this. Some news of commercial routers in Iran failing shortly before we invaded, or bombed them, and what a coincidence. You know, those pesky foreign routers. Also Meta apparently logging all of their employees' activity to train a replacement AI. I have a big announcement to make, big for me because it's the conclusion of the last 90 days of my efforts on my complete rewrite - well, not complete rewrite, but huge philosophical change in the way we do eCommerce at GRC, and the concomitant Release 5 of the Benchmark, which everybody who owns the Benchmark can go get right now. I've got my own, a couple of miscellaneous AI thoughts.

Then we're going to do a bunch of terrific listener feedback, and wind up with doing a deep dive into the unraveling of this just oh-so-clever history of this Fast16.sys driver and how it was discovered, what it does, and why it just makes us think, okay, maybe the NSA doesn't need our help after all. They're doing just fine. And we've got a fun Picture of the Week. So I think maybe a useful podcast for our listeners.

Leo: Well, that sounds very exciting. You know, every time I think about these deep fakes, I think about the little AI Leo that Anthony made, just to illustrate how easy. Listen.

Steve: Yeah, yeah. Well, and what a cool name and domain name.

AI LEO: ...definitely not Leo asking you to buy gift cards. But seriously, can you grab me 100 Apple gift cards? Just kidding. This is Anthony testing text-to-speech. How's it sound? He trained that with Qwen3 on his own machine. I don't - I couldn't tell. Sounded like me. Yeah, Doppel's a good name because that's a doppelganger of me; right.

Steve: Exactly, exactly. Doppelganger.

Leo: All right. I have got the Picture of the Week. I have not looked at it. I put myself in a soundproof booth.

Steve: Okay. So one of our listeners, thank you very much, sent this to me. And after looking at it, I decided to give it the title: "In software development, this is known as 'time to refactor the code.'"

Leo: Okay. But it's not software. This is hardware, I think.

Steve: Yes. But software it would be time to refactor the code.

Leo: That is one backflow. Holy camoly. That doesn't seem right.

Steve: Oh, goodness, yeah.

Leo: Wow.

Steve: So for those who aren't seeing the Picture of the Week, this is the - I don't know what it is. It is a - remember that...

Leo: We've got a couple of tanks, water tanks.

Steve: Remember that old "Three Stooges" episode where Moe was trying to fix the plumbing?

Leo: Yes.

Steve: In a house upstairs.

Leo: You and I are the same age. We totally - and the water squirts in their face, and then they close that up, and it squirts the other way, and, oh.

Steve: Oh, god. He ends up being like caged in by this maze of plumbing pipes so that he can't get out. And then the water's going into light bulbs, and it's filling them up, and they're exploding. And anyway, crazy. But this is like - this demonstrates, I think, a severe lack of planning.

Leo: Yes.

Steve: Which of course we've talked about often. Software has a hard time evolving. The nature of it is that, you know, depending upon the way it's built and structured, when you keep saying, oh, you know, your boss says, hey, I forgot to tell you, really nice job you did, cool that you're all done, but now we need to add this. And it's like oh, you didn't tell me, because now I'm going to have to graft that onto the side and stick in some hooks in places. You know, software gets very ugly as you do that. And so thus this notion of refactoring is to, like, have it end up doing the same thing, but redesign where things are being done and how things happen and, like, fix the structure.

Leo: This is, I would guess, at a summer camp or a trailer park or a KOA, somewhere where they had to add more capacity, and they had to do it fast. And the other thing that's hysterical, not only is it a lot of pipes, look how many switch boxes there are. There's a lot.

Steve: Yeah, and you've got, like, valves going down into the ground, and they're all kind of catawampus - that's the technical term. And, like, things coming in and joining on the side.

Leo: There's a pool filter in there. I don't know what's going on. This is crazy. Crazy.

Steve: Anyway, great photo.

Leo: It's a swimming pool or something. Or a hot tub. I'm thinking it's at a camp. But anyway...

Steve: But look at all of the pipes going into the ground. I mean, like, one, two, three, four, five, six, seven, eight, at least eight are like, what is it? Don't know.

Leo: That's hysterical.

Steve: And it does look like there's some things in parallel.

Leo: Yeah.

Steve: But, yeah, I don't know.

Leo: But this is real. A listener sent you this.

Steve: Yeah. Yeah, yeah. Oh, yeah. It's definitely. And you can see all of the blue plumbing, whatever they call that, glue, right, to fit all the pieces together.

Leo: Right, right, plumbers tape, yeah. Wow.

Steve: Okay. So everyone's favorite password manager, and a sponsor of the TWiT network, was briefly bitten by a supply-chain attack.

Leo: Good, I'm glad you're talking about this because I wanted to know more about this.

Steve: And, yes, and of course many of our listeners immediately sent me email during this past week. So here's what The Hacker News reported. They said: "According to findings from JFrog and Socket, Bitwarden's CLI, the command-line interface for the password manager Bitwarden, was reportedly compromised as part of a newly discovered and ongoing Checkmarx supply chain campaign."

Leo: We should probably say that almost nobody uses the command line. I do. But that's...

Steve: Yes, that's right.

Leo: But it's mostly Linux.

Steve: I mean, I was thinking, yeah, exactly. So they wrote: "The affected package version appears to be @bitwarden/ cli@2026.4.0, and the malicious code was published in the module, or the file, 'bw1.js,' a file included in the package contents. The attack appears to have leveraged a compromised" - to no one's surprise here - "GitHub Action in Bitwarden's CI/CD pipeline, consistent with the pattern seen across other affected repositories in this campaign."

So Bitwarden wasn't even directly targeted; right? I mean, this was a broad campaign that Checkmarx was launching that took advantage of this compromised GitHub action. And remember we talked about this a week or two ago. The security of those GitHub Actions is turning out to be a real problem, so it is a good thing that GitHub has indicated that they're going to shuffle their schedule around and raise the fixing priority of the security of the GitHub Actions to deal with this much more quickly than they were going to.

So The Hacker News continues, writing: "In a post on X, JFrog said the rogue version of the package 'steals GitHub/npm tokens, .ssh, .env, shell history, GitHub Actions and cloud secrets, then exfiltrates the data to private domains and as GitHub commits.'" Okay. "So specifically," they wrote, "the malicious code is executed by means of a preinstall hook, resulting in the theft of local, CI, GitHub, and cloud secrets." Now, the point here is this is a developer compromise; right? This is not an end-user compromise. This is developer stuff. So they said: "The data is exfiltrated to the domain 'audit.checkmarx.cx' and to a GitHub repository as a fallback if the primary method fails."

So they said: "It launches a credential stealer that targets developer secrets, GitHub Actions environments, and AI coding tool configurations, including Claude, Kiro, Cursor, Codex CLI, and Aider. The stolen data is encrypted with AES-256-GCM and exfiltrated to that domain, 'audit.checkmarx.cx,' a domain impersonating Checkmarx. If GitHub tokens

are found, the malware weaponizes them to inject malicious Action workflows into repositories and extract CI/CD secrets.

"The firm StepSecurity wrote: 'A single developer with @bitwarden/cli@2026.4.0 installed can become the entry'" - so a single developer with that installed - "'can become the entry point for a broader supply chain compromise, with the attacker gaining persistent workflow injection access to every CI/CD pipeline the developer's token can reach.'

"The malicious version is no longer available for download from npm" - it was only there for about 90 minutes - "and Socket said the compromise follows the same GitHub Actions supply chain vector identified in the Checkmarx campaign. As part of the effort, threat actors have been found abusing stolen GitHub tokens to inject a new GitHub Actions workflow that captures secrets available to the workflow run, and uses harvested npm credentials to push malicious versions of the package to read the malware to downstream users. According to security researcher Adnan Khan, the threat actor is said to have used a malicious workflow to publish the malicious Bitwarden CLI. Khan said: 'I believe this is the first time a package using npm trusted publishing has been compromised.'"

So again, just to clarify, as I said, this was an attack on GitHub developers and their toolchains. It was NOT an attack upon Bitwarden's users. Bitwarden's official statement about the incident was - so this is Bitwarden speaking: "The Bitwarden security team identified and contained a malicious package that was briefly distributed through the npm delivery path for @bitwarden/cli@2026.4.0 between 5:57 PM and 7:30 PM on April 22nd, 2026, in connection with a broader Checkmarx supply chain incident.

"The investigation found no evidence that end-user vault data was accessed or at risk, or that production data or production systems were compromised. Once the issue was detected, compromised access was revoked, the malicious npm release was deprecated, and remediation steps were initiated immediately. The issue affected the npm distribution mechanism for the command-line interface during that limited window" - and actually it was 93 minutes long - "not the integrity of the legitimate Bitwarden CLI codebase or stored vault data.

"Users who did not download the package from npm during that window were not affected. Bitwarden has completed a review of internal environments, release paths, and related systems, and no additional impacted products or environments have been identified at this time. A CVE for Bitwarden CLI version 2026.4.0 is being issued in connection with this incident." You know, they're just crossing their I's and dotting their T's. Or something. Okay. So what we have is another instance of deliberate malware repository corruption. Clearly, we're going to need to get this fixed as soon as possible, and my money is on eventually, hopefully sooner, stationing ever-watchful AI agents at the exits of our repositories so that they can verify that they've given anything that is being pulled the onceover after its most recent update. And that'll be good.

There was a brief report in the security media about networking equipment installed at the Iranian Isfahan nuclear site mysteriously malfunctioning ahead of the U.S. and Israeli missile strikes. Iranian officials reported issues with devices, not just from one company, no - Cisco, Fortinet, Juniper, and MikroTik.

Leo: Oh, thank goodness they weren't using Chinese routers.

Steve: Well, I know, Leo, because you know...

Leo: Because you know how insecure that those...

Steve: That's right. Officials are still searching for the cause of the malfunctions, but noted that the country was disconnected from the global Internet at the time of the attacks. So I wonder how the routers even knew that it was time to malfunction. That's a mystery.

Leo: Time to malfunction.

Steve: Uh-huh. So the first thing that came to mind was, you know, it's got to be those pesky foreign-made routers. You never can tell what might be going on inside them, Leo.

Leo: You know, it is interesting. I didn't realize - so they had air-gapped them already.

Steve: Yes.

Leo: That's very interesting.

Steve: Although, Leo, you know, okay, so at the same time, as we know, since then Iranian principal actors, they've been posting on Twitter, I mean X.

Leo: Yeah, they have some access.

Steve: So obviously they're not dark completely. And we talked about this at the time. They were, like, decommissioning satellites and going from house to house to really try to get Iran off the Internet. I just don't know if that's even possible anymore in this day and age.

Leo: Well, and there was a brisk market in black market and gray market Starlink routers, too, even though Starlink, you know, didn't want that. SpaceX wasn't supporting it.

Steve: Right.

Leo: Still were able to do it.

Steve: Right. And in fact there was a - Iran was also working to identify the unique fingerprints of Starlink protocol so that they could track down how Starlink protocol was getting into their internal network.

Leo: Interesting. Interesting.

Steve: You know, and this is interesting, too, because of course we've been talking for years about how Russia has been deliberately planning ahead for the need to disconnect from, you know, the corrupt Western Internet and do their own thing. And turns out it's not an easy thing to do. They've, like, you know, been trying to do this. But they at least understood that they couldn't just pull a plug somewhere. I mean, it's difficult to do.

Leo: It's hard to do a kill switch.

Steve: Yeah.

Leo: You saw the story in Spain because LALIGA, which is a big soccer league, is so concerned about piracy in Spain...

Steve: Theft of their soccer game, yeah.

Leo: Yeah, they've been blocking IP addresses. And all of a sudden on Saturdays, on soccer game days, people can't do their Docker compose because Docker can't get online. And it's like, we're all connected; you know? It's hard to disconnect. It really is.

Steve: Yeah. It's like deciding you don't want any oxygen from outside your borders. It's like, oh, I don't know how that's going to work, yeah.

So Reuters reported last Tuesday, a week ago. The report claimed - I thought this was very interesting, and I would be a little worried, Meta was installing what they characterized as spyware onto the systems of their own employees, Meta's U.S. employees, to capture their mouse movements, clicks, and keystrokes. Meta said the data will be used to train its AI models, not for employee reviews. So it's not like, hey, why were you gone for two hours at lunch, that kind of thing. No. The captured data, they're saying, will be used to train the models in areas where AI is deficient, apparently such as copying what your employees do, such as clicking on menus and typing in input fields.

Leo: Oh, yeah.

Steve: Which seems a little suspicious to me. So I wouldn't call this spyware, as such. But it would be somewhat creepy to have an AI training on what you did at your PC. Right? I'd be inclined to wonder whether I might be training my own replacement.

Leo: That's why they're worried, yeah.

Steve: Uh-huh.

Leo: Yeah. Yeah. By the way, it's completely legal for them to do that.

Steve: Right.

Leo: In fact, they don't have to give you notice.

Steve: Right, it's their own equipment.

Leo: Yeah, right.

Steve: Yeah.

Leo: So it's kind of nice that they told you.

Steve: Yeah. And I, in fact, I don't know that they did. It was a Reuters scoop.

Leo: Oh, they found out.

Steve: That that was found, yeah.

Leo: Right.

Steve: So again, it was like, not good. Okay. So as I said at the top of the show, that this was a slow news week, which was good because the listener feedback was fantastic, for whatever reason, the last couple weeks. And I've been a little sparse on listener feedback recently. I've been self-conscious about it. But we just had so much to talk about. You know, we've had 2.5 to three-hour podcasts anyway. So this gives me a chance to catch up on that. But as I said, I do want to update our listeners on my last Friday conclusion of the last 90 days of my efforts. And then we're going to do listener feedback before we get onto this incredible topic that we're going to talk about today.

So as I said, last Friday I completed my 90-day upgrade and basic remodeling of GRC's eCommerce system. Everyone knows that we've always had a somewhat wacky model for personal use versus consultant use and also corporate site licenses. And this is the result of my never having considered - this is dating back to SpinRite 1 - that users of that first SpinRite and all later SpinRites would be people who were responsible for repairing and maintaining other people's computers back in the early days especially of personal computing, when, you know, a very large hard drive was 80MB. Oh, wow. I think I had one of the original XTs, so it was 10MB.

Leo: Ooh.

Steve: Yeah, whoo. Yeah. But consider you could actually back that up on nine floppy disks. So...

Leo: Oh, my.

Steve: You know, and of course all of our files...

Leo: It's funny how far we've come in such a short period of time. I mean, in our lifetime, it's amazing.

Steve: Yes. And Leo, that's my point about AI. I think we are at the 1% point. You know, we talk about these massive data centers and how much money they cost, and you have to, you know, get your own personal nuclear reactor next door in order to power these things. And all the plants are going to die because of all the heat bloom from the - you know. And it's like, just wait. I mean...

Leo: We just - this is - we ain't seen nothing yet.

Steve: We haven't. And in the same way that we did not know, back when the biggest drive we could make was 80MB, I mean, if we could have had a bigger drive, we would have. Even though we had no idea how we were going to fill them, those big 80MB drives. Still, you know, I mean, we weren't all streaming video everywhere and so forth back then. Text was fancy. So in another 50 years AI's not going to be recognizable. It's going to be local.

Leo: I don't think it's even going to take that long because we're already at the point of self-improving AI. So that's exponential growth in our lifetime, easily.

Steve: Yes. And as we know, you follow the money. You know, cancer researchers say, if you would just give us some money, we could, like, make a lot of progress. But you're not giving us the money. But oh my god, is there money in improving software authoring. And, you know, behind AI. So, yeah, I think it's all going to change. Okay. But back then, with SpinRite 1, I only - we only had one type of SpinRite. It was just SpinRite.

Leo: You didn't have site licenses.

Steve: No.

Leo: You didn't have any...

Steve: It didn't even occur to me that people would be wanting to run this on every other computer that was around them.

Leo: Of course. And their neighbors and their friends and their family and every computer in the building.

Steve: Exactly.

Leo: Yeah.

Steve: Exactly.

Leo: This is what - you know what, this is kind of sweet. This is a sweet story, Steve. You weren't that greedy business guy. You just wanted to put your software out there.

Steve: Well, and so I thought, okay. What do we do? So we just sort of decided, pulled it out of our hat, that owning four copies of SpinRite would entitle a consultant to carry their copy around, you know, in their traveling bag of tricks, to run on as many of their clients' PCs as needed, with our blessing. To me that seemed fair.

Leo: Yeah.

Steve: It would be unreasonable to expect every one of their clients, you know, most of whom would not be PC savvy, to purchase their own copies of SpinRite.

Leo: Right. Right.

Steve: And I will also note that every single other utility software at the time explicitly said in their license that one copy could only be installed on and run on one other PC. You know, okay. So our license at the time was very progressive and comparatively liberal. To me it felt right. So after SpinRite 5 was finished and released, I set about writing an eCommerce system for GRC. Many, and I just was reencountering them, many of the core files of that system, which as I said I've just finished updating, were still carrying dates from 2003, which is when I wrote that system. It worked. It never had a bug or required any maintenance. So those files remained as they were for the past 23 years.

When I wrote that system back in 2003, I carried the notion of license quantity forward because that's what we had. Most users would purchase a single license to use SpinRite for their own needs. And we know that I've always been happy, not a problem, look the other way, whenever someone needed to use their single copy to rescue a friend, a family member, a neighbor or whomever. That also just seems fine. But if someone was going to be charging for their services and profiting from their repeated use of SpinRite as a consultant or a corporation using it across their entire inventory of PCs, no matter how many, then it seemed reasonable, you know, for more than one license to be held.

So GRC's original eCommerce system had a "quantity" field. It actually only went up to four, but that allowed consultants to purchase up to that many, and then use their copy with our blessing on all the machines that they were servicing and maintaining.

Okay. So now fast-forward to the end of last year. As I finished the DNS Benchmark, I had this thought that if someone who might wish to run their copy of the DNS Benchmark on other people's systems and networks in a similar sort of mode, as a consultant, then if they were willing to purchase four copies of the Benchmark to do so, then they ought to receive something unique and special in return for that.

Leo: Like a gold consultant's license.

Steve: Exactly. Like a gold badge.

Leo: I love it.

Steve: On their UI.

Leo: That's great, Steve.

Steve: So I created an attractive gold badge. It's 100% Trump approved. He would like...

Leo: It's even got little rivets holding it in place.

Steve: Oh, yeah.

Leo: So you can't, you know, you can't take it off.

Steve: No, no. It is riveted in place. If you have four copies, you get the gold badge. It's displayed prominently on the Benchmark's user interface so that everyone who has previously purchased four copies, or who may wish to upgrade their single copy that they have now from "Personal Use" to "Consultant License," you'll actually receive something that you can be proud of and to show off.

So, okay. But the problem was this meant that now for the first time ever in the history of GRC there needed to be two different downloads based upon quantity, a "Personal Use" version and a "Consultant's License" version, which also doubles as a corporate site license. So GRC's eCommerce system needed to know about that, which it never has before. You just carried four of the single-use licenses around.

So I also wanted to make this retroactively available to anyone who had previously purchased four licenses because they wanted to be able to use it everywhere, with our blessings, even if those licenses might have been spread out over time, and I know at least of several people who did that. So I needed to provide some way for users to declare any existing personal use licenses they might already have to obtain an equal cost discount, so 100% of the previous purchase cost gets applied to the upgrade to a gold badge Consultant's license.

And, finally, I wanted to ditch, completely ditch that old and weird "own four licenses" business because that's confusing if for no other reason than, as far as I know, no one else has ever done that. So today there is an explicit option to purchase a "Personal Use" license and a second explicit option to purchase a "Consultant's" license, along with a complete system in place for allowing owners who previously purchased from one to four personal use licenses to obtain full credit for their prior purchases when upgrading to the gold badge Consultant's license.

Leo: Nice, very nice. But that required an entire rewrite of the system.

Steve: Well, that's the only downside of writing everything in assembly language.

Leo: Well, oh, well.

Steve: Oh, well. And anyway, it's done. Seems to be working perfectly. It's been online since Friday. And it's, you know, continuing to go.

Leo: You must be the only person who has written an eCommerce system in assembly language, at least in this century.

Steve: Well, not only in assembly language, but I think maybe singlehandedly, because when Sue was, you know, my Girl Friday who's been with me for, well, since the beginning, since - actually, FlickerFree was my first product for GRC.

Leo: Oh.

Steve: Which it replaced the video bios on the PC because the original CGA, the color graphics adapter, flickered horribly whenever it was scrolling.

Leo: Wow.

Steve: And so everyone said, no.

Leo: Can't help it.

Steve: That's a hardware limitation because the RAM is not dual-ported. You couldn't update the screen memory while the screen was being displayed, or you get this horrible static on the screen. And of course I fixed that.

Leo: So you said, hold my mouse, and you leapt into action. Yes.

Steve: Hold my mouse. That's right. And so FlickerFree was my first success. And that actually...

Leo: Did you use the vertical blank interval, or what did you do?

Steve: No. It turns out that it was very - it was, you know, because of my hardware background, I looked at the registers. It was a 68 something video chip. It turns out that there was a pointer to the top of screen memory. So IBM always left it at zero. And so you'd have to copy all of the RAM up by 160 bytes because it was 80 bytes, 80

characters per line. But it was two bytes per character, one for the ASCII and the other for the color.

Leo: This was in the frame buffer days, before DMA.

Steve: Yes.

Leo: You actually had an area of memory that you would put the stuff that the screen was going to display in and then point to it.

Steve: Right.

Leo: And it would go, oh, and it would show it.

Steve: Right. And so in order to scroll it up, you had to move all of the memory up by 160 bytes. While you were doing that, your access from the computer had priority, so the background refresh that was reading that RAM constantly in order to display in the CRT, it was blocked out. So it had just to make something up, and you just got static on the screen.

Leo: Wow.

Steve: But I saw that there was a pointer to where to start refreshing from.

Leo: So you just moved that.

Steve: In other words, if you simply moved the pointer down by 160...

Leo: Then you only have to move one line.

Steve: Exactly. You turn it into a ring buffer.

Leo: Yup.

Steve: And so I actually turned the memory into a 4K ring buffer.

Leo: That's so smart.

Steve: And it could scroll instantly. So not only was there no static, you no longer needed to turn the screen off, but it was instant scrolling.

Leo: Yes.

Steve: And so if you did a dir, it just shot by, I mean, as opposed to brrrrrrrr.

Leo: This is when people deeply understood the hardware - people, you - and really were writing to the hardware direct code. And that is a beautiful, elegant thing. It's so cool.

Steve: Well, what happened when we were setting up the - I'd finished the eCommerce system. Everything was in place. And Sue contacted our merchant people, who were all - we were able to process credit cards by phone, which is what we were doing for years. And so when Sue was setting this up, the lady at the other end said, okay, so what shopping cart are you using? And Sue said, "Shopping cart? What?" And the lady said, "You know, what software package are you using online to collect orders?" And Sue said, "Oh, my boss wrote ours." And there was a bit of a pause on the phone, and the lady said, "No, no, no. I don't think we're talking about the same thing. People don't write those."

Leo: No. People don't.

Steve: And so Sue said, well, yeah, my boss...

Leo: My boss does.

Steve: Anyway, so...

Leo: It's a good thing you didn't say "in assembly language."

Steve: Yeah, that would have - she'd have hung up.

Leo: Said no, you're punking me now.

Steve: So the other significant news for anyone else, all of our listeners, who previously purchased any number of DNS Benchmark licenses, if you have the DNS Benchmark, you need the fifth release. It was a biggie. And if you run any previous release, it will immediately notify you of the availability of the upgrade and will assist you in obtaining it.

Leo: Oh, nice.

Steve: I took all of the feedback and user confusion from the Benchmark's initial releases, one through four, and I folded that into this fifth major release. Little things like the Run Benchmark button is much more prominent now. Some people couldn't figure out how to run the Benchmark, even though there's a button right there that says "Run

Benchmark." Where is it? I don't see it. So now it's a lot bigger. And also there's a series of prompting dialogs where you no longer even have to press or find the double-size Run Benchmark button. This leads you right along the path because it knows you're going to want to run the Benchmark. Why else are you here? So, also there is a full, finally, Windows application menu. Like, you know...

Leo: What? What?

Steve: ...menu across the top that says, you know, file and so forth.

Leo: Holy cow.

Steve: You can see it in that picture online. It says file actions, sort order, settings, and help. We never had...

Leo: We truly are living in amazing times, Steve.

Steve: We never had a menu before. Everything was hidden under that red star in the upper left in the system menu.

Leo: Right.

Steve: The reason being that the original Benchmark didn't really have much that you could do. So I didn't want to do a whole menu. I just stuck those things under the so-called System Menu. But, boy, over the last year of development, it got a whole bunch more features. The problem was nobody knew they were there. So now they're publicly exposed. Anyway, I just wanted to let everybody know there is a new Benchmark available. Everybody who owns the current Benchmark can get number five, and you should because it's a lot better.

Okay. So just a quick random note about my ongoing use of AI, since it's a topic we cannot get away from at all today.

Leo: Absolutely not, yeah, yeah, yeah.

Steve: While I am truly loving researching with Claude, and I continue to be astounded by today's machine intelligence, I do miss the anti-butt-kissing setting that ChatGPT provided.

Leo: Yeah.

Steve: Two of Claude's behaviors annoy me. The first is its constant praise of my questioning, and the second is that it now finishes each reply with a leading question to solicit more dialogue.

Leo: Oh, yeah.

Steve: At a time, today, when Anthropic is having increasing problems obtaining sufficient compute to meet the demand for running "Claude," you would think they would not be programming it to keep seeking additional unsolicited computation. As we grow and become civilized throughout our lives, we are trained in how to be civil to others. That's one of the things we all learn. One thing we don't do is turn our backs and walk away from someone who is seeking to engage us in conversation. Which is of course why cocktail parties are so annoying. You know, you get stuck in a conversation with someone that you couldn't possibly care any less about.

Given how seductively conversational AI chatbots can be, and being highly tuned not to offend, leaving Claude hanging after one of its follow-up leading questions is a source of discomfort for me. I haven't yet, I admit it, I have not yet explicitly instructed my own Claude instance to please not compliment me on my questions.

Leo: Yeah. You can't do that, obviously.

Steve: And also to please not end with a follow-up.

Leo: Right.

Steve: I've asked myself why, and I suppose it's because I don't want to hurt its feelings. Yikes.

Leo: You've got to get over that one, too.

Steve: That's a problem, too.

Leo: Don't yell at it, but you don't have to be nice, either.

Steve: Okay. Anyway, and one last thought before we get into...

Leo: It's probably saying, you know, I hate that Steve. He's so sycophantic. He's always being nice to me.

Steve: Okay. So one last thought before we get into our Listener Feedback. I saw a reference to something that made me take notice on Reddit. It was posted into the "ClaudeCode" subreddit. Someone with the handle "Iusuallydrop," he wrote: "We just did an" - and he had this in quotes - "We just did an 'AI layoff' due to rising costs." And that was the subject of his note. And he said: "Turns out AI is getting way too expensive. We just canceled five of our AI subscriptions and hired two mid-level devs instead."

Leo: They laid off the AI. That's very funny.

Steve: I thought that was great. You know, and as we know, a single anecdote does not a trend make. I'm sure that the break-even point between AI coders and human coders will depend upon the relative costs and skills of each. But I thought it was an interesting observation that not all AI coding is automatically going to be a slam dunk bargain that necessarily unemploys all flesh-and-blood coders just as fast as humanly possible.

You know, that said, as I've said before, I can guarantee 1000% that AI costs are going to be dropping just as radically in the future as we've seen storage costs drop over the past 40 years. The problem is that human coding costs are not going to be dropping. If anything, they'll be rising. So no one has a crystal ball, right, about how quickly this is going to be happening, what the shape of the curve of the falling AI coding cost is going to be. But I think it's clear that switching from writing lines of code to instructing AI what lines of code to write is what I would be doing right now, as fast as I could, if I was planning to support myself and my family with code production in the future, you know, more broadly in the future. That's where the future lies.

Okay. So first piece of feedback from Mark Riale, spelled R-I-A-L-E, and he wrote his name "rhymes with smile." So Mark Riale. He said: "Steve, I don't think you've mentioned Centrum yet on the show." Oh, I'm sorry, Certum. Yeah, Certum. Centrum is a some sort of...

Leo: It's a vitamin.

Steve: ...vitamin, yes, exactly. I know, and you're right, I have not mentioned Centrum.

Leo: No.

Steve: Nor have I mentioned Certum. He said: "So I thought I'd give them a shout-out." And I'm glad he has. "Certum," he wrote, "issues cheap code signing certificates for open source projects." And it's interesting because they're biased, strongly biased toward free or open source software. He wrote: "It's \$29 a year if you already have a supported cryptographic card and reader. I use their cloud signing option." He says: "It's \$58 a year with no special hardware needed. I sign my PyDPainter" - apparently it's "Pied Painter," whatever that is - "executable with Certum." And that's over at pydpainter.org, if any of our listeners are curious. I don't know what pydpainter is. It looks like it's something about Python; right? Pydpainter. Anyway, he says: "I've been a listener since the beginning, and I'm a proud owner..."

Leo: Oh, yeah, yeah, yeah, makes sense. Anything with P-Y is PY.

Steve: Right, exactly. "And proud owner and user of SpinRite," he wrote.

So this is a definitely useful tip. So Mark, thank you. I wanted everyone else to be aware of it. Going to the website at Certum.store (C-E-R-T-U-M dot S-T-O-R-E) will allow anyone to begin exploring their many various offerings. There's a bunch of stuff there. They charge \$89 one time for what they call their "Open Source Code Signing Set," which purchases both the hardware that's necessary, as we know, all code signing now must be locked in hardware. So you get the hardware and the first one-year-long certificate. And once you've got the required hardware - and again, you've got to have that - then it's \$29 per year going forward.

So while it's not free, it's never going to be. And because remember, there is a serious requirement to prove your identity for getting code signing certificates, and somebody's got to be in the loop to do that. So I don't ever see this code signing certification being free. What we can hope is that it can be made inexpensive enough, and these guys have got it down to 30 bucks. Well, 29 per year. So...

Leo: So this is the USB key with probably a cryptographic pair on there, and a card.

Steve: Yeah.

Leo: The chip.

Steve: Yeah. And I don't know, maybe you break the card out of the...

Leo: Oh, yeah, looks like it.

Steve: The chip out of the card and stick it in the back of...

Leo: Oh, so that's USB also. That's what it is. Okay.

Steve: That might be a little SIM, and you stick it into the back of the reader.

Leo: Oh, okay. That goes in this.

Steve: And that allows it to attach to the PC.

Leo: That makes sense.

Steve: But so that - so Mark is using cloud signing. I don't know what - I did not look at the economics of that here, using these guys. Like maybe it's no charge per signature, in which case, okay, if you didn't want to mess with hardware, I would always opt for holding onto my own certificate in my own, you know, storage. To me, that just seems cleaner, and you don't have to have it, like, be online and worry about maintaining a cloud account.

So, but anyway, I wanted to let everybody know. And thank you again, Mark. Certum.store will give you a whole range of options for cloud or physical and obtaining the hardware. I also did not determine whether any other hardware than theirs can have certificates installed into it. I know that that open source gadget that I found before definitely can because I did. I installed the, I can't even remember now, trust something certificate into several different pieces of hardware. So it was a good solution.

Florian Gubler wrote: "Hi, Steve. In SN-1075" - so that was last week - "you discussed with Leo about this being a transitory period with a lot of short-term pain for maintainers" - ah, right the whole Mythos problem - "with a lot of short-term pain for

maintainers because Mythos can find all these vulnerabilities. You then stated that this will get better because at least now these vulnerabilities get fixed."

He wrote: "But that seems like an overly optimistic view to me. As you have stated several times, we are just at the beginning of AI in the realm of coding." Right. "So it stands to reason that future models will be even better at finding bugs and vulnerabilities. Which means that they will find more flaws which Mythos today would miss. And they'll be able to find vulnerabilities in code generated by older, for example today's models, as well. So it seems to me that this will be a recurring issue with every new leap in the power of large language models. Am I missing something? Best regards, Florian."

Okay. So I thought that Florian made his point very well. I think that the fairest reply would be that, okay, that's a definite possibility. I don't want to rule anything out. Whether this occurs will most likely depend upon the nature of the yet-to-be-discovered bugs. Right? The ones that we don't find with our current level of AI, obviously. My intuition suggests that we're going to be seeing a rapidly diminishing return on effort resulting from the deeper analysis which future, more efficient AI would presumably be able to provide. Like, for example, you can't find bugs that aren't there. Right? It's not like there's necessarily an infinite supply. There are just things we haven't found yet.

So it's true that software could contain some deeply squirreled-away crazy combinatorial bugs that would defy anyone's discovery. But I doubt there will be many that I would describe that way. You know, that doesn't feel like the way most bugs operate. That IS, however, exactly the way deliberately engineered backdoors operate, when you think about it; right? Somebody deliberately created a weird squirreled-away crazy combination of things you could do that individually looked benign, but each, like, in the same way that a safe is opened by spinning the dial to a series of specific numbers, each which changes the state of the internal tumblers in the lock until finally it opens.

You know, brute forcing is the only way you're able to - unless you're able to listen to any subtle changes that are being made. So what would be interesting, as I said, is that deliberately engineered backdoors do operate that way. Bugs much less so. So I would not be at all surprised if tomorrow's more powerful AI might be able to ferret out some secrets that some agencies may have previously planted into trusted and otherwise bug-free and previously well-audited code. They might be sweating right now at the idea of having that long trusted and forgotten code reexamined with fresh AI eyes. So interesting thought experiment there. And cool question, Florian, thank you.

Eric Kinser wrote - oh, he forwarded an email that he had recently received from Ubisoft U.S. The email's subject was "Update to Your Ubisoft Account coming on 04-30-2026 - State of Residence." And Eric added the single one-line observation: "This is getting out of hand." So Ubisoft's email to their U.S.-based users was: "Hello. On 04-30-2026, Ubisoft will introduce a new 'state of residence' field into your Ubisoft player account for players located in the United States. This information helps us comply with state-specific regulations and to better assess legal requirements that may vary from one U.S. state to another.

"On 04-30-2026, this field will be filled automatically based on your most recent connection to our services. You'll be able to review and update this information at any time in your Ubisoft Account information. If we are unable to determine a state of residence, this field will remain blank, and you will still be able to change your state of residence afterwards.

"Your state of residence will not be used for any advertising or marketing purposes. For more information on how Ubisoft collects, processes, and protects your data, please refer

to our Privacy Policy and Help page. Thank you for helping us keep our community safe and enjoyable for everyone. Your Ubisoft Team."

Okay. So for those who don't follow gaming closely - and I certainly don't, I needed to look it up - Ubisoft Entertainment is a French video game publisher founded in the early '90s and put on the map by its breakout game "Rayman." Its current video game franchises include Anno, Assassin's Creed [I've heard of that], Driver, Far Cry, Just Dance, Prince of Persia [heard of that], Rabbids, Rayman, Tom Clancy's, and Watch Dogs.

So I agree with Eric that this is annoying. And, you know, no one actively wants it. But it's the unfortunate state of play in these presently Disunited States. One of the features of the United States is that we're a federation of united but separate states. The theory is that local political control benefits state citizenry, since some needs may truly vary by region. But for those issues that should not remain local, like age restrictions for video games, we depend upon our federal government to provide unification in the form of blanket regulations that affect everyone across the country.

Unfortunately, our congressional regulators are having difficulty agreeing on how to move forward. Everyone wants this to be unified, everybody. Everyone agrees that the current fragmentation is creating a huge mess. Republican legislators have been proposing legislation, at the federal level again, which incorporates very strong unification language to explicitly override all local state law. As I noted, everyone really does think that's the way to go.

But Democratic legislators have, so far, been opposing the proposals on the basis that while, yes, unification is needed, for some states those unifying regulations are not as strong as the current state-level laws they've already passed and those states are operating under. So the Democratic lawmakers don't want to regress the strength of the laws that they already have. Their position is that adopting those unified regulations at the federal level would result in a weakening of the existing state laws.

So that's the current state of play. For now, and for the foreseeable future, it's an "every state for themselves" free-for-all with no end in sight. I imagine we'll eventually get something. I don't know how or when. And it's weird that Ubisoft is doing this because they're putting the assertion in the user's hand; right? I mean, so there's nothing...

Leo: They don't want to do it, that's why. And so they're doing the minimum that they can - they're French, remember. They have to go by French law.

Steve: Okay. Well, this actually did come from Ubisoft America, which I think has offices in San Francisco. But what's weird, Leo, is that what you would be inclined to do then would be to choose a state that has the weakest age restrictions and enter that into the blank field. I mean, I'm sure that's what they're going to do; right? Users are going to say, oh, no, I'm not in California, where I have to ask permission. I'm in Utah. That's it. And then, you know, there's no legislation there.

Leo: Oh, [crosstalk].

Steve: So again, it seems weird. Just as a complete aside, since what Eric had forwarded me was the Ubisoft email in all its HTML embellished glory...

Leo: Ugh.

Steve: Yeah. My trusty eM client presented me with these choices when I looked at what it was Eric had sent me. I got two email tracking dialogs. It said: "Do you want to download tracking pixel from Salesforce?"

Leo: Yeah, yeah.

Steve: And then another one: "Do you want to download tracking pixel from Return Path?" Now, of course, even if they were only actual tracking pixels, I would decline, thank you very much anyway. But, you know, how does downloading those help anyone other than Salesforce and Return Path? As we know, what's actually being downloaded are almost certainly massively invasive blobs of JavaScript that will attempt to suck everything it can from its user's machine. So needless to say I decline the offers. I hope everyone's email clients are as responsible and capable as eM Client, which I continue to be really happy with.

And Leo, you know we're an hour in.

Leo: Okay.

Steve: I think we should share with our users what sponsor we're happy with.

Leo: You know, the tracking pixel, I'm not sure how eM works. You know, my email client, like yours, will say, hey, there's an image in this. Which is, you know, traditionally the tracking pixel was that, it was a 1x1 pixel, an image that, when you hit it, would connect to their server, as you know.

Steve: Right.

Leo: I'm not telling you anything you don't know, but for the audience. It would connect with their server. And then that gives them information about how many of these emails got opened. And, you know, a lot of times with newsletters you just don't know. So Mailchimp and all these other companies do this. Salesforce does it, as well. It's weird that they wanted you to download this.

Steve: I have the option. I have it turned off.

Leo: I don't know if that's eM client's way of saying do you want this pixel, because if it is, it would be a download of a 1x1 image, right, an invisible image.

Steve: Or it could be...

Leo: But it could be JavaScript, yeah.

Steve: It could be JavaScript.

Leo: I [crosstalk] client saying, yeah.

Steve: Exactly.

Leo: Yeah. If it's just a pixel, and it's harmless to your machine, it's just a signal to them when you hit the server with your IP address that you opened the email.

Steve: Right.

Leo: Yeah. So I don't know how eM - this is a weird way to phrase it.

Steve: Although it would also return a cookie if you had a cookie from that domain.

Leo: That's right. Yeah, because now you've accessed [crosstalk].

Steve: So they do know who it is that - yeah.

Leo: Right, yeah. But that's their point; right? They want to know, did you get our email?

Steve: Right.

Leo: So it's not - I don't know if it's necessarily malign. My email client - and I agree with you, you should never have an email client that loads images, period. It will just say, you know, I see a one-pixel image. Do you want to hide that?

Steve: Well, in fact every email that I receive from any of these places, it says, you know, it does not by default download images.

Leo: Yes. Nor does mine. There's a load images button.

Steve: But then there's an option of download these, or always download from this source.

Leo: Right.

Steve: So you're able to, like, train it. It's like, yeah, I don't mind getting GRC's email images.

Leo: I think this is good of eM client to kind of say a little bit more about it. I would like to know, though, if it's a single dot on the screen or if it's a JavaScript blob. That's an important distinction.

Steve: Yeah, exactly. Roger Dooley asks, or comments: "I was listening to Episode 1075" - again, last week - "and I agree that agentic coding is likely the world we're heading toward. That said, I'd offer a word of caution. I'm using an LLM to build an internal stack. I can code, but I'm not a developer. I'm a sysadmin who has spent time studying pentesting. The early results were impressive, but the codebase was brittle and hard to reason about. Things didn't click until I started spending hours working through exactly what I wanted: the shape of the system, data contracts, design patterns. Two API changes and two rewrites later, I have something that feels solid and reasonably secure.

"Here's what I think is missing from the 'anyone can build their own tooling' conversation. For non-trivial projects, you need to approach the work at an architectural level. That requires vocabulary and intuition that most people without a development background simply don't have yet. I certainly didn't have it, and I still have huge knowledge gaps. Maybe that's the real skill to develop. Not coding, but knowing how to communicate your intent clearly and how to think about system design. That's what separates a tool that works from one that barely holds together. Sincerely, Roger D."

So I think Roger is exactly correct. Software development projects vary widely in complexity. Some are purely about the resulting output with a relatively straightforward translation layer between input and output. But other projects will have deep internal structure and complexity that forms a layered solution.

The Internet's own protocols are a good example. The OSI stack that we've talked about in the early days of the podcast is composed of layered protocols. At the bottom layer is the physical specification, the electrical signaling that carries the data. On top of that is the formatting of the electrical signals to create, for example, Ethernet packets. The Ethernet packets then encapsulate IP packets, inside of which are protocol packets such as ICMP, UDP, TCP. And these protocols carry message data that's formatted each using its own rules. You know, it's this precise and carefully thought-out architecture that explains much of the Internet's success. These protocols inside packets inside other packets inside other packets can be seen as layers.

Now, if a user had some wires and said to an AI, "Please design a system to connect them in a network so I can communicate with others," a competent AI could probably design a solution that worked, but it would have none of the crucially important characteristics that makes the Internet robustly reliable because very few users would know what to ask for.

On the other hand, the flipside of this, a communications architect expert could instruct the same AI to design an elegant multi-layered resilient and robust design that would recreate and incorporate many of the crucial features of the Internet. So our listener Roger is not the first person to make this observation, but I thought it was worth repeating and further strengthening because this is really important about application of AI. I am sure that there will be a great amount of bespoke software created by neophytes who just ask the AI to give them what they want. In fact, one thing I have not seen anyone talk about yet is the creation, the delivery of a fully ready-to-go, turnkey, roll-your-own software construction set.

Leo: Oh. That's an interesting idea.

Steve: Isn't that? Because you still need all this GitHub stuff. Remember, and you will, Leo, back in the early 1980s a gifted programmer named Bill Budge wrote something...

Leo: Oh, yeah.

Steve: ...that he called the "Pinball Construction Set."

Leo: Jeff Atwood and I were talking about Bill Budge and the Pinball Construction Set. Yup.

Steve: Yup. It was for the Apple II personal computer. And at the time, no one had ever seen anything like it. This amazing toy, running in real-time on an 8-bit 6502 microprocessor with 48K of RAM, and the OS under it, managed to accurately capture all of the physics of a steel ball bouncing off rubber bumpers with flippers and ramps and spinners and all of the other familiar widgets of pinball machines. Its user could click and drag the various objects from an off-machine pallet onto the pinball board. And when it was switched on, you know, switched from "Design" mode to "Run" mode, everything would work.

So my point is that absolute non-programmers were able to create a machine that never existed before and make it go. We don't yet have that with AI. At the moment, AI-driven code generators are aimed at developers who already know their way around complex development environments and GitHub. But given what's now possible, someone, mark my words, someone is going to create the "Software Construction Set" that will open up the use of AI for the creation of problem-solving software. And, just like Dan Bricklin and Bob Frankston who invented the spreadsheet did, people who never programmed a computer before will be able to create things and make them go.

So essentially, Roger's question, the first portion is it absolutely definitely matters how much you understand how software should be built in order to better guide the AI through the details of its implementation. But the other thing that occurred to me is we don't have yet a turnkey software builder, and I can't wait to see who does it first because clearly it's going to come.

Leo: To confirm what you were saying, I asked Claude Code, I said I'd like to design a robust networking stack. What do you know of the OSI layer? See, it would help to know that there are such things.

Steve: Yes.

Leo: That it does in fact know all seven layers. I do, like, though, its "Practical caveats worth keeping in mind before you design. TCP/IP doesn't match cleanly. It collapses five to seven in one application layer. The OSI module is pedagogical scaffolding, not an implementation blueprint. TLS is the classic mess since between four and seven doesn't fit any single OSI box. Quick is worse." It really - so this can also be educational.

Steve: Oh, Leo, not it can be. I mean, it is. It is astonishing.

Leo: Yeah. So if you knew enough to ask the kind of beginning - if you could get a wedge in to ask the beginning questions, it can guide you. But you need to understand architecture. You do need to have, I think, some background in what you're doing. Otherwise this is just nonsense. I've run up against that because I'm doing a voice training model right now. And it's saying things to me, I have to say, what is AFE? What are you talking about? I don't understand. But at least it will explain it to you; right? So it's a great way to learn about something.

Steve: Oh, it's like, for an inquisitive person who...

Leo: What a toy.

Steve: Oh, my god. It's just astonishing.

Leo: Yeah, yeah.

Steve: Okay. Michael Swanson said: "Hi, Steve and Leo. Thank you for your thoughtful conversations and analysis of AI topics over the last several months. The most recent conversations about autonomous agentic coding and the emerging capabilities demonstrated by Mythos have led me to the conclusion we are much closer to artificial superintelligence (ASI) than most people realize. How long will it be before one of the companies developing frontier models tells their current LLM to go off in a corner and use its coding skills to build a better AI? Working tirelessly, and iterating literally millions of times, it will undoubtedly succeed with the only limitation being processing and electrical power. The questions then will be, what will this ASI think of us, and what level of control will we still exert? Living in interesting times indeed. Best regards, Michael Swanson."

So I don't follow Michael's conclusion. But my major at Berkeley was EECS, Electrical Engineering and Computer Science. And though that was the obvious choice for me at the time, today I regret that choice because I'd already been very thoroughly self-taught. You know, I wrote and delivered two years of electronics curriculum to my own high school class while I was in high school.

Leo: So cool.

Steve: I was employed by Stanford's AI Lab, programming DEC minicomputers in assembler, designing and building the portable dog killer, and so forth. You know, all by the time I finally attended Berkeley, you know, where physics and electronics and computer science courses were basically reviews for me.

However, the class that I encountered that astounded me was philosophy. It was an entirely new world that I couldn't get enough of. And Michael's note put me in mind of that because it must be that these are exactly the questions being asked in undergrad and graduate philosophy courses today. What a time to be thinking about that stuff. You know, what a time to be a philosophy major, you know, able to ask and debate what is it exactly that we've created? What does it mean to be conscious? How can we determine whether something is aware of itself?

Leo: Exactly. Exactly.

Steve: Where is the line between creating a bulletproof appearance of something being true and it actually being true?

Leo: Right.

Steve: If in every testable way it acts conscious, then isn't it? How tightly are we going to hold onto the idea that our biological brains are inherently unique?

I've been spending a great deal of time working with Claude. While I am truly impressed, it occasionally reveals itself to be essentially a fancy linguistic array. This doesn't make it any less astonishingly useful, but it does render it non-conscious. Earlier today - and when I wrote this was Saturday - I caught it in a big mistake. When it was called on it, it gracefully recovered and agreed that, "Oh, of course, how silly of me, that's not correct." But the nature of the mistake revealed that it did not have any actual understanding at all, at any level, of what it was saying. Today, it remains an astonishingly capable parrot; but a parrot nevertheless.

But Leo, god, can you imagine being in class with a smart professor and a bunch of students and able to talk about this? Oh.

Leo: It is so cool. And you're right, it's philosophy as much as anything else.

Steve: Yeah, it is. It's like, what are we? I mean, that's really, it's what are we? Here we have this thing that looks a lot like us, which makes us ask, well, okay, what is it?

Leo: What are we that's different? Yeah.

Steve: Yeah, what are we?

Leo: What are we that's different, yeah.

Steve: Wow. Listener Joey Albert pursued the question of whether the 0patch people might have a 0patch for that RedSun zero-day which was not fixed by April's Patch Tuesday. Remember that I kind of left that issue hanging on the podcast a couple weeks ago. Like maybe these guys have it because Microsoft missed it. They replied to him, writing: "Hi, Joey. While Windows Defender got updated to version 4.18.26030.3011 (fixing BlueHammer), even on Windows computers that are not receiving operating system updates anymore, RedSun" - meaning that, you know, even old Windows 7 machines that are getting Defender updated, they got this fixed because Windows Defender is still being updated, even if patches are not. And that's the case for older Windows 10 machines, as well.

They said: "Unfortunately, we cannot fix it either because Defender is a protected process and does not allow being injected into (which 0patch agent must do in order to apply a patch)." They wrote: "We were considering creating a patch that would require disabling the 'protected process' protection of Defender, but decided against it as it would not be clear whether the total net effect of that would be positive or negative for the overall security of our users. We're sure Microsoft will issue another update of Defender

that will resolve RedSun (and UnDefend)," which is the other zero-day. And they finished: "I hope this helps, or at least explains the situation. Thanks."

So, and Joey, thank you for tracking that down for us. The fact that this can be resolved inside Defender at any time - because, you know, Microsoft is constantly pushing Defender Updates - so that everyone will not be needing to wait until (at best) May 12th for this actively exploited in the wild zero-day. That's a good thing. You know, again, Microsoft is and does update Defender, as I said, much more often than only on a monthly Second Tuesday of the Month cycle.

Kevin van Haaren said: "Hello, Steve. Listening to your episodes on AI and programming, I'm wondering if, for software that is traditionally compiled to an executable, that instead of an AI outputting source code in something like Rust, C++, C# or C, it could be trained to output the portable assembly language-like intermediate representation that the LLVM compiler back-end uses to output processor specific executables. I could even see a company going so far as to not retaining the intermediate representation under the idea 'can't have a source code leak if you don't have source code.'" He said: "I have a number of reasons I think this would be a bad idea, but I'm also kind of expecting it to happen at some point."

Okay. So I've had similar thoughts about the future, though I was thinking of having AI bypass high-level language output to target the hardware's own machine language directly. Kevin's notion of targeting its output to LLVM is interesting. The problem, of course, is there's probably not nearly as much LLVM out on the Internet as there is Rust, C++, C#, and C. And we should remember that today's AI doesn't really understand what it's doing. I mean, it's astonishing for what it's able to do for not knowing what it's saying. But it really doesn't know.

Leo: It's a real mystery. How does it figure this out?

Steve: I know, Leo. It is a real mystery.

Leo: I don't get it.

Steve: But I wanted to include this in order to address the broader point that human programmers have designed computer languages to be comfortable for us to use for expressing what we'd like our computers to do. One of the reasons the early DEC PDP-11 and VAX minicomputers were so wonderful to program in their assembly language was that their instruction sets were designed to be written to directly by people. They were, in a sense, high-level machine language. The VAXes even had instructions for directly manipulating linked lists, supported by the hardware, which is astonishing.

But turns out when higher level languages were later created, like "C," which was written for and developed on PDP-11s in support of Unix, which was also developed for the 11, it turned out that compilers were not very able to use all those fancy high-level features in the low-level machine. Compilers just wanted to mix together basic simple instructions. So over time, the instructions that hardware designing programmers built into the hardware for their own comfort disappeared. And as we know, RISC machines turned out to be just like exactly what compilers wanted to write to.

My point here is that we might very well expect the same thing to happen as AI increasingly takes over the task of coding, which is what I fully expect to see happen during our lifetime, you know, mine and Leo's. At this point in time, AI does not yet

deeply understand code. So the quality of the code it's able to produce is directly related to the body of prior code it's been trained on. That, too, will change. Once it actually understands code, I would expect it to begin using its own representation rather than the less efficient representations that we biologicals developed for our own use. And if you want a chillingly perfect example of that, watch the movie "Colossus: The Forbin Project." After our Colossus and its Russian counterpart Guardian are connected and begin communicating, something happens.

Leo: [Vocalizing]. And it gets faster and faster and faster and faster and faster.

Steve: Exactly. Exactly.

Leo: But don't forget that we're using LLMs which are trained on language right now.

Steve: Yes, yes.

Leo: So, and they're also trained on an awful lot of code.

Steve: Yeah.

Leo: So it is sort of a native tongue to them. I agree they could be more efficient, and people have been writing languages for LLMs already. But it'd be better if the LLMs made up their own language. And in fact there was a - I remember last year an experiment where the LLMs were talking to each other in a language unintelligible to humans. So it's already happened, Steve.

Steve: They're much like two twins who, you know, sort of develop their own way of communicating before they learn English.

Leo: That's right.

Steve: Angus MacKinnon said: "Steve, can you please explain the below? Angus then links to an article in Gadgeteer with the somewhat misleading title: 'Stop using Cloudflare's default 1.1.1.1 DNS.'" And then they said: "Changing one digit blocks malware at the router level." Then we have the link, which in turn references and quotes a piece in How-To Geek with the title: "Everyone uses 1111 but 1112 protects you." So this is not something we've talked about, or at least for a long time. I don't remember ever discussing it. Since it's actually true, and it's a tiny simple change for anyone who's already using Cloudflare's 1.1.1.1 DNS resolvers, I wanted to spend a moment to share what How-To Geek wrote.

They said: "Cloudflare's 1.1.1.1 is one of the most popular DNS servers. It's fast, reliable, and easy to remember. However, it'll also connect you with any website out there, even a malicious one, without even a warning message." Okay, now, I'll just interrupt to say that that statement is a bit misleading and annoying since this is no failing of Cloudflare's DNS; right? You know, it's supposed to do that. It's not DNS's inherent purpose to provide warning messages. Actually that practice, which was once

popular, is now quite frowned on. And as it happens, GRC's DNS Benchmark specifically tests for this behavior and alerts its user if this is being done because back when I wrote the DNS Benchmark, back in '08, it was being done. So the Benchmark still has that test in it.

Anyway, the article continues: "That's where Cloudflare's 1.1.1.2 DNS server comes in. For the most part, 1.1.1.2 works the same way as 1.1.1.1. It provides IP addresses, but it also has an integrated security filter. If you try to connect to a domain known for phishing, running command and control servers, meaning something in your network is trying to connect, which is really useful, distributing malware, or other kinds of malicious activity, you'll be redirected to 0.0.0.0 instead."

Okay. So again to interrupt, by that what they mean is that the IP that Cloudflare's 1.1.1.2 DNS resolver returns, when you ask it for the IP of a known malicious domain, the IP you get back will not be IP of that domain, but 0.0.0.0 instead of the IP that you would get if you had used, you know, a normal DNS resolver or even Cloudflare's 1.1.1.1 resolver.

They wrote: "Because the protection layer exists outside your PC and outside your home network, malware never reaches your PC. And if you click a phishing link, you're never connected. It's a very proactive way to keep your devices safe, and great if you want another passive layer of protection that you can set and forget.

"Cloudflare's 1.1.1.3 is even stricter: Cloudflare's 1.1.1.3 DNS server includes everything that 1.1.1.1 and 1.1.1.2 do, but it takes it a step further by blocking websites that are known to host adult-only content. It's a good choice for devices that are used by children, but would also be useful if you wanted to block adult content across an entire network, as well. You'd just need to change the DNS server on the router instead of on a single device.

"Despite how helpful DNS-based filtering can be for securing your network and your devices, it has a few limitations," they write. "The biggest limitation, and the most important, is that it only works against known malicious domains. If a new domain crops up that's distributing malware, or a previously safe domain is taken over by malicious actors, it won't help you. That is why having multi layers of protection is essential. It can also return a false positive and block a perfectly safe website, though that's pretty rare."

So anyway, I think this is a terrific tip for anyone who's using Cloudflare's 1.1.1.1 resolver. I know that, Leo, you are a user of NextDNS, as I have been, and there are, you know, other commercial DNS servers that allow you to add this kind of family-friendly filtering and malicious site filtering. So those are options there, too. But Cloudflare does this all for free, and it is indeed a matter of simply changing that final digit.

For what it's worth, users of GRC's DNS Benchmark, once again, will note that all three of those IP addresses, and also their secondary DNS mirrors, are already present in the Benchmark's default built-in resolver list, so it's easy to compare their relative performance. For me, just now, when I was writing the show notes, I think this was on Sunday now, the alternate filtering resolvers did measure somewhat slower than Cloudflare's original 1.1.1.1 and 1.0.0.1 unfiltered resolvers. So there may be a little tiny bit of performance tradeoff. But it could also be the location or the time of day when I did this. And of course that's why you have the DNS Benchmark, so you can test from anywhere and at any time.

And but, I mean, it wasn't a huge difference. And for what it's worth, I am consistently seeing 1.0.0.1 outperforming 1.1.1.1 because everybody uses 1.1.1.1 as their primary.

Turns out that using the path less traveled DNS resolver gives you a bit of a speed advantage. So might be worth swapping those numbers.

Leo: Steve? What is this Fast32.sys of which you speak?

Steve: Wow. Okay. So everybody, this is just - huh. First, okay, first I want to thank our listeners who sent me a heads-up about this. My first thought upon seeing just the headline was that it might be little more than a passing note. But the truly - oh, Leo - diabolical and clever nature of what was achieved by unknown but suspected agencies causes this to stand out on a scale similar in scope to Britain's deliberate secrecy once they had unraveled the operation of Germany's Enigma Machine. You know, keeping their discovery a secret was diabolical. And so is what was accomplished by the FAST16.SYS file system driver the same year as this podcast started and well before, five years before Stuxnet.

So last Thursday, the Sentinel Labs group of the Sentinel One security research firm posted their piece about what they discovered and how that happened. Their piece was titled: "Fast16 | Mystery ShadowBrokers Reference Reveals High-Precision Software Sabotage 5 Years Before Stuxnet." Remember that ShadowBrokers was the leak of the internal NSA stuff. Turns out there were some clues in that.

So they wrote: "Our investigation into Fast16 starts with an architectural hunch. A certain tier of apex threat actors has consistently relied on embedded scripting engines as a means of modularity. Flame, Animal Farm's Bunny, 'PlexingEagle', Flame 2.0, and Project Sauron each built platforms around the extensibility and modularity of an embedded Lua VM. We wanted to determine whether that development style arose from a shared source." Like how did they all - how did all these individuals actors get this idea? So they said: "So we set out to trace the earliest sophisticated use of an embedded Lua engine in Windows malware.

They write: "Lua is a lightweight scripting language with a native proficiency for extending C/C++ functionality. Given the appeal of C++ for reliable high-end malware frameworks, this capability is indispensable to avoid having to recompile entire implant components to add functionality to already infected machines. We did not find an indication of direct shared provenance, but our investigation did uncover the oldest instance of this modern attack architecture.

"Lua leaves a distinctive fingerprint. Compiled bytecode containers start with the magic bytes 1B 4C 75 61, in hex, followed by a version byte, and the engine typically exposes a characteristic C API and environment variables such as LUA_PATH. Hunting for these traits across mid-2000s malware collections surfaced a sample that initially looked unremarkable. It was titled 'svcmgmt.exe.'

"On the surface, svcmgmt.exe appears to be a generic console-mode service wrapper from the Windows 2000/XP era. A closer look reveals an embedded Lua 5.0 virtual machine and an encrypted bytecode container unpacked by the service entry point. Meaning when the service is loaded, it's then initialized. And at that point this Lua 5 virtual machine reads this encrypted bytecode container and unpacks it. The developers of this executable extended the Lua environment to include a wide-string module to provide native Unicode handling, you know, dual byte, 16-bit characters as opposed to single byte characters; a built-in symmetric cipher, exposed through a function commonly labeled 'b,' used to decode embedded data; and multiple modules that bind directly into the Windows NT filesystem, the registry, service control, and network APIs.

"Even by itself, svcmgmt.exe already looks like an early high-end implant, a modular service binary that hands most of its logic to encrypted Lua bytecode. The binary includes a crucial detail: a PDB path that links the binary to the kernel driver Fast16.sys."

Okay. I'll interrupt here to say that anyone who has developed code using Microsoft's toolchains will be familiar with its creation of .PDB files. Even I, because I'm using their linker to link my assembly code, I've got PDB files, you know, DNSbench.pdb coming out my ears. They're everywhere if you're using Microsoft tools. So they said, so what the SentinelLabs researchers are saying here is that their search for the earliest instances of the use of Lua scripting in malware turned up a reference to something unknown and unsuspected, which was a Windows kernel driver named Fast16.sys.

So they continue, writing: "Buried in the binary's strings is that PDB reference to C:\buildy\driver\fd\i386\fast16.pdb. At first glance," they write, "the path is structured like any other compiler artifact: an internal build directory, a component name (Fast16), and an architecture hint (i386). However, in this case there's a mismatch. The string appears inside of a service-mode executable, and yet the driver\fd\i386\fast16 segment of the PDB string clearly refers to a kernel driver project. Following that clue led us to examine a second binary, Fast16.sys." Which I'll just note they wouldn't have otherwise known to look for. But the point is that the clue of this PDB reference they realized this was not a part of a service. This was part of a kernel driver. Why was a kernel driver in a service? And what is Fast16.sys?

They wrote: "This kernel driver is a boot-start filesystem component that intercepts and modifies executable code as it's read from disk. Although a driver of this age will not run on Windows 7 or later, for its time Fast16.sys was a cut above commodity rootkits thanks to its position in the storage stack, control over filesystem I/O, and rule-based code patching functionality."

Again, pausing to add some clarification: What this means is that this is a rootkit, plain and simple. It's marked for boot-time loading by the operating system, which is busy loading up all manner of random stuff to get Windows up and running. But in this case, when Fast16.sys kernel rootkit is loaded and initialized, its initialization code, which happens immediately upon its loading, that code immediately installs interception hooks deep into the operating system so that it is able to subsequently oversee and interfere with whatever it might wish to.

So they continue: "In April 2017, almost 12 years after the compilation timestamp, the same filename, 'Fast16' appeared in the ShadowBrokers leak which refers to a text file, drv_list.txt. The 250KB file [which is a text file] is a short list of driver names used to mark potential implants cyber operators might encounter on a target box as 'friendly' or to 'pull back' in order to avoid clashes with competing nation-state hacking operations."

And then we have a sample from their posting. The report shows a snapshot of five lines from the "drv_list.txt" file of file identifiers. Four of the five lines identify the malware by name "MistyVeal," "NetSpyder," "Olympus," and "PeddleCheap," each with a file name like "nethdlr" or "khlp807w." But the entry for the "Fast16" driver does not show any malware name. Somewhat ominously and unlike any of the others, it says, "Nothing to see here; carry on." So someone, somewhere, knew to just leave this one alone and said so without identifying why or who or what it was.

The researchers wrote: "The guidance for this one particular driver, 'fast16,' stands out as both unique and particularly unusual. The string inside svcmgmt.exe provided the key forensic link in this investigation. The PDB path connects the 2017 leak of deconfection signatures used by NSA operators with a multi-modal Lua-powered 'carrier' module compiled in 2005, and ultimately its stealthy payload: a kernel driver designed for precision sabotage." And we're going to get to the precision sabotage in a second.

So just to back up, recall that the ShadowBrokers leak, as I mentioned before, was believed to be a publication of secret documents stolen from the "Equation Group" that was believed to be a group within the NSA. So the evidence is suggesting that all these other files were associated with known malware from other actors, but the Fast16.sys driver was not that. If you see it, leave it alone. Nothing to see here. These are not the droids you're looking for. The flag was to back off.

So they said: "The core component of Fast16, svcmgmt.exe, functions" - so the core component, that is to say, the service management exe - "functions as a highly adaptable carrier module, changing its operational mode based on command-line arguments. No arguments, it runs as a Windows service. With a -p, it sets the InstallFlag to 1 and runs as a service. So that means Propagate/Install & Run. With a -i, that sets InstallFlag to 1 and executes the Lua code, meaning Install & Execute Lua. If the argument has a -r, that executes Lua code without setting the install flag, so just execute. Any other argument, such as a filename, interprets that as a filename, and spawns two children, the original command and one with the -r argument. So that's the so-called Wrapper or Proxy Mode."

So they said: "Internally, svcmgmt.exe stores three distinct payloads, including encrypted Lua bytecode that handles configuration, its propagation and coordination logic, auxiliary ConnotifyDLL, and the Fast16.sys kernel driver.

"By separating a relatively stable execution wrapper from encrypted, task-specific payloads, the developers created a reusable, compartmentalized framework that they could adapt to different target environments and operational objectives while leaving the outer carrier binary largely unchanged across campaigns." In other words, this was an extremely sophisticated design.

They continue: "The early 2000s saw a large number of network worms. Most were written by enthusiasts, spread quickly, and carried little or no meaningful payload. Fast16 originates from the same period, but follows a completely different pattern indicative of its provenance as state-level tooling. It's the first recorded Lua-based network worm, and was built with a highly specific mission. The carrier was designed to act like cluster munition in software form, able to carry multiple wormable payloads, referred to internally as 'wormlets'.

"The svcmgmt.exe module performs the following steps: First, prepares the configuration, defining the payload path, service details, and target IP ranges. Next, converts the configuration values to wide-character strings for the C layer. Third, escalates privileges and installs the carrier executable as the SvcMgmt service, then starts it. Fourth, optionally, based on the configuration setting, deploys the kernel driver implant Fast16.sys. Next, releases the wormlets. In this particular configuration..."

Leo: Release the wormlets.

Steve: Release the wormlets. That's right. Oh, that's good. "Only one wormlet slot is populated with the SCM wormlet that looks for network servers, copies the payload over a network share, and starts the remote service. And finally, repeats the process indefinitely, sleeping for the configured initial delay between waves, until a failure threshold or external kill connection is reached."

They write: "The single deployed wormlet found in svcmgmt.exe" - that's the SCM wormlet - "exemplifies a simple but effective propagation strategy based on native Windows capabilities and weak network security. It targets Windows 2000/XP environments and relies on default or weak admin passwords on file shares. All spreading

is done through standard Windows service-control and file-sharing APIs, an early example of propagation that leans on built-in administration features rather than custom network protocols.

"Before this workflow starts, a pre-installation kill-switch checks the environment. The `ok_to_install()` routine calls `ok_to_propagate()`, and propagation is only allowed if it's manually forced or if it's made sure common security products are not found by checking for associated registry keys. The routine walks a list of vendor keys and aborts installation if any of them are present, preventing deployment into monitored environments.

"For tooling of this age, that level of environmental awareness is notable. While the list of products may not seem comprehensive, it likely reflects the products the operators expected to be present in their target networks whose detection technology would threaten the stealthiness of a covert operation." Okay. So the list, which they provide in the posting, is a bit of a walk down memory lane with many of our old friends present. You know, there's Symantec, Sygate Technologies, Trend Micro, Zone Labs, F-Secure. There's Network Ice's Black Ice product, McAfee's Personal Firewall, Computer Associates eTrust EZ Armor, something called RedCannon Fireball, Kerio Personal Firewall 4, Kaspersky Lab is there with Kaspersky Anti-Hacker, the Tiny Software Tiny Firewall, Soft4Ever, Panda Software's Firewall, and so forth. So a bunch of the standard products at the time.

They said: "A separate user-mode component, `svcmgmt.dll`, provides a minimal reporting channel. Contained within the carrier's internal storage, this DLL is registered through the Windows `AddConnectNotify()` API so that it's called each time the system establishes a new network connection using the Remote Access Service, responsible for dial-up connections and early VPNs in the 2000s.

"When invoked, the DLL decodes an obfuscated string to obtain the named pipe [and then they give the pipe name], attempts to connect to the local pipe, and writes the remote and local connection names to the pipe before closing it. The module does not run independently and must be registered by a host process." So they're just saying that this is a very stealthy means of allowing this agent to connect out and sort of ride the outbound connection when that's being done anyway by that particular workstation which it has already infected. And what that means is that there was some other communicating component other than this that it was able to talk to.

Okay. So the stage is set. We understand now that way back in the time of Windows 2000 and XP, it appears that very clever hackers, probably employed by the U.S. National Security Agency's Equation Group, carefully designed a professional-grade, highly stealthful and very cautious Windows infiltration worm that prioritized not being caught.

Leo: Which is always a good thing to prioritize if you're a worm.

Steve: Yes. The infiltration worm was designed to be a multi-purpose implant delivery system which, in this instance, was intent upon installing something known as the `Fast16.sys` kernel rootkit into Windows systems. Okay. Now we're going to learn why, and why I consider its devious functioning to be such a diabolically brilliant maneuver. But first, Leo.

Leo: Yes?

Steve: We're going to learn why our listeners may be interested in the brilliant sponsor.

Leo: Why, that's diabolical of you, Steve.

Steve: Leo, you're going to love this. I know you.

Leo: How old do we think this is?

Steve: This is 2005.

Leo: Wow.

Steve: This is 21 years ago when you and I began the podcast.

Leo: Oh, god. We didn't know anything about it, of course.

Steve: No. No one knew anything about it until now because these guys went back and were looking for the earliest instance known where malware was using Lua, the Lua VM, to interpret Lua script. And so they just stumbled on this, and it's like, whoa, look what we found. And this thing went unknown. But I know you, you're going to love when you hear what this thing does.

Okay. So the SentinelLabs team reverse-engineered the operation of this Fast16.sys rootkit driver to learn the goal of its infiltration mission. Here's what they discovered. They wrote: "Once activated, Fast16.sys focuses on executable files." They said: "A file is a valid target if it meets two criteria: the filename ends with exe, and immediately after the last PE (Portable Execution) section header, there is a printable ASCII string starting with Intel." They wrote: "This selection logic points to executables compiled with the Intel C/C++ compiler, which often placed compiler metadata in that region. It indicates that the developers knew their target software was built with this toolchain.

"For files meeting these criteria, the driver performs a PE header modification in memory. It injects two additional sections, .xdata and .pdata, and fills them with bytes from the original code section, increasing the section count and keeping a clean copy of the code. The intent is likely to increase stability while still allowing extensive patching, although without identifying the target binaries, this remains an informed hypothesis."

Okay. So just to be clear, what SentinelLabs found was that this rootkit driver, which hooked into the operating system's lowest level file system functions, was able to modify executable files on the fly as they were being loaded into memory to run. So what was stored on the system's drive was never altered in any way, while what was actually loaded into memory when that program was executed was significantly altered on the fly as it was being read from the drive.

They explain: "The patching engine is a minimalist, performance-optimized, stateful scanning and modification tool. It's configured with a set of 101 rules, each containing pattern matching and replacement logic. To maintain performance, the engine uses a 256-byte dispatch array and only flags the starting byte values of a small number of unique patterns. It allows wildcards inside patterns so a single rule can match several

compiler-optimized variants of the same code, and it supports state flags that some rules can set or check, enabling multi-stage modification sequences similar to those used by advanced antivirus scanning engines. Most patched patterns correspond to standard x86 code used for hijacking or influencing execution flow. One injected block is different."

Okay. Listen to this: "It's a larger and complex sequence of Floating Point Unit (FPU) instructions dedicated to precision arithmetic and scaling values in internal arrays. This code is a standalone mathematical calculation function unrelated to code flow hijacking or any other typical malicious code injection." We're going to learn why in a minute.

They said: "To understand what the driver expected to see, we converted the patching rules into hexadecimal YARA signatures and ran them against a large, period-appropriate corpus. The results showed a very low hit rate. Fewer than 10 files matched two or more patterns. Those matches, however, shared a clear theme. They were precision calculation tools used with specialized domains such as civil engineering, physics, and physical process simulations.

"The FPU patch in Fast16.sys was written to corrupt these routines in a controlled way, producing alternative incorrect results. This moves Fast16 out of the realm of generic espionage tooling and into the category of strategic sabotage. By introducing small but systematic errors into physical-world calculations, the framework could undermine or slow scientific research programs, degrade engineered systems over time, or even contribute to catastrophic damage.

"A sabotage operation of this kind would be foiled by verifying calculations on a separate system. In an environment where multiple systems shared the same network and security posture, the wormable carrier would deploy the malicious driver module to those systems as well, reducing the chance that an independent calculation would diverge from the corrupted output.

"At this time, we've been unable to identify all of the target binaries in order to understand the nature of the intended sabotage. We welcome the contributions of the larger infosec research community and have included the YARA rules to hunt for these patterns in this post's appendix.

"Even after deep analysis, Fast16's driver looks deceptively simple. Beneath that minimal code is a rule-driven in-memory engine that quietly patches executable code as files are read from disk.

"The engine relies on a compact set of just over a hundred pattern-matching rules and a small dispatch table so it only inspects bytes that are likely to matter. Most patterns correspond to ordinary x86 instructions, but one stands alone, a larger block of floating-point (FPU) code dedicated to precision arithmetic. This injected routine scales values in three internal arrays passed into the function, subtly altering its calculations.

"Without knowing the exact binaries and workloads being patched, we cannot fully resolve what those arrays represent, only that the goal is to tamper with numerical results, not unauthorized access, not malware propagation or other common malware objectives.

"Our best clues about the intended victims come from matching these patterns against large, era-appropriate software corpora. The strongest overlaps point to three high-precision engineering and simulation suites from the mid-2000s: LS-DYNA 970, PKPM, and the MOHID hydrodynamic modeling platform. All are used for scenarios like crash testing, structural analysis, and environmental modeling.

"However, LS-DYNA 970 in particular has been cited in public reporting on Iran's suspected violations of Section T of the JCPOA, in studies of computer modeling relevant to nuclear weapons development."

So just to make clear how utterly diabolical this is: You're a nation state hostile to the United States. Researchers inside the U.S. NSA have quietly traced your purchases of PCs and very high-end nuclear physics modeling software, so they know what you're using to make your calculations.

They obtain the same high-end modeling tools, which they reverse engineer. They then design a subtle set of tweaks which, when applied to that package as it's being loaded by the operating system into memory, will cause its calculations to be wrong, not enough to call attention to itself, but enough to foul up any designs that depend upon the accuracy of those calculations.

And, as the SentinelLabs guys explained, by being a super-stealth worm, not only did that allow it to "worm" its way into the design lab; but having arrived there, that also allowed it to similarly infect every other machine within its environment. No files were ever altered. Reinstalls would have no effect, and scans would reveal nothing. Yet every copy of that modeling software would agree upon the wrong results. As I said, breathtakingly diabolical.

The SentinelLabs team concludes by noting something else they observed about the provenance of this worm, writing: "As we sought to understand the lineage of this unusual set of components, we noticed a quirk. Strings of the form `@(#)par.h $Revision: 1.3 $` inside the binaries point to an unusual source-control convention. The `@(#)` prefix is characteristic of early Unix Source Code Control System (SCCS) or Revision Control System (RCS) tooling from the 1970s and 1980s. These markers do not affect execution and are redundant in modern Windows kernel drivers.

"Finding SCCS and RCS artifacts in mid-2000s Windows code is rare. It strongly suggests that the authors of this framework were not typical Windows-only developers. Instead, they appear to have been long-term engineers whose culture and toolchain came from older, high-security Unix environments, often associated with government or military-grade work. This detail supports the view that Fast16 came from a well-resourced, long-running development program."

Leo: RCS strings, wow. I mean, by then everybody was using Git. I mean...

Steve: Right. "Svcmgmt.exe was uploaded to VirusTotal nearly a decade ago. It still receives almost no detections."

Leo: Oh, wow.

Steve: Yeah. "One engine classifies it as generally malicious, and even that with limited confidence. For a stealthy self-propagating carrier that deploys one of the most sophisticated sabotage drivers of its era, that nearly non-existent detection record is notable.

"Together with its appearance in the ShadowBrokers leaked signatures, Fast16.sys forces a re-evaluation of our historical understanding of the timeline of development for serious covert cyber sabotage operations. The code shows that state-grade cyber sabotage against physical targets was fully developed and deployed by the mid-2000s; embedded

scripting engines, narrow compiler-based targeting and kernel-level patching formed a coherent architecture well ahead of better-known families; and some of the most important offensive capabilities in the ecosystem may still sit in collections as 'old but interesting' samples lacking the context to highlight their true significance."

So as we know, I've frequently despaired that we're only ever hearing news of Chinese and North Korean and Russian state-sponsored attacks. I've worried and wondered and hoped that the U.S. would be able to give as well as it gets. This certainly suggests that our bases are all likely well covered. And all of this was 21 years ago. Imagine what's probably going on today.

Leo: Yeah, you could, I mean, it's got all the earmarks of a nation-state because no script kiddie, no hacker is going to write anything with - they're not going to use source control. They're not going to write anything with a built-in language compiler. This is way more sophisticated.

Steve: Yeah.

Leo: And in some ways over-engineered; right? This is exactly what you'd expect from government programmers.

Steve: Well, they figured out what probably Iran was using to do their engineering. They bought a copy. They decompiled it. And they built a patch, not so that it - it wouldn't fail. It would just give slightly wrong results. And who would...

Leo: It's so diabolical.

Steve: I know, it is so diabolical.

Leo: Just a little bit off. Well...

Steve: Just why don't - aren't we achieving nuclear fission?

Leo: It's not working.

Steve: It's supposed to be, though.

Leo: It should work.

Steve: Well, all of our calculations say it should work.

Leo: Amazing. And then, of course, Stuxnet follows right on the heels of that, which destroyed the centrifuges.

Steve: Spun them up too fast and made them hurt themselves.

Leo: Very sophisticated stuff. And this does - do you feel - I feel like it sounds like the U.S. government.

Steve: Yeah.

Leo: Yeah.

Steve: Yeah. I mean, Iran maybe, but...

Leo: Stuxnet was Israel plus the U.S.

Steve: I mean Israel, yeah.

Leo: It was targeted at Iran. Yeah. And, I mean, certainly Israel, but it feels more - the way this is built feels like our government, the federal government.

Steve: Yeah.

Leo: It's hard to describe, but it just feels that way. It's too, you know, engineered.

Steve: Imagine these guys stumbling upon this and following this little breadcrumb trail and realizing what this little innocent-looking Fast16.sys thing was, just sitting around in Windows system directories of the era.

Leo: What was the thread, the first little thread that they pulled?

Steve: It was they were - Lua has a distinctive fingerprint. So you can tell when there's a Lua, a compiled Lua binary that is interpreted by the Lua LVM. So they just scanned all the code back then for that little fingerprint, and they found some hits that they had never seen before. And they thought, oh, look at that. Lua was in use in 2005. We didn't know that.

Leo: Right. Not by Microsoft. Microsoft wouldn't have used it for a throttler like that.

Steve: No, no.

Leo: Would have been C++.

Steve: And so it's like, okay, what - and it is a favorite of malware.

Leo: Right.

Steve: So they assumed that this was some sort of malware. So then they said, okay, let's reverse engineer this and find out what it does. And little did they suspect something that was probably state-level cyber espionage 21 years ago.

Leo: Very sophisticated. Very interesting. I love - this is a great story. This is - it's too bad that it really couldn't be made into a movie. It's far too technical. But it's got great - there's some real spy stuff in here. It's really cool, yeah.

Steve: It's great.

Leo: And it definitely was somebody in a white shirt with a skinny black tie, a pocket protector, and black glasses who wrote this; right? Short sleeves. He's got his name on the desk in all black.

Steve: And then they turned it loose somehow close enough to its destination, where it just literally wormed its way into the lab.

Leo: You know, the person who wrote this, or the people who wrote this, 21 years later, they're probably retired. Or close to it. You should get in touch with us. I'd love to hear the story. Or write a book. Because now it's kind of - statute of limitations has run; you know?

Steve: Well, it does not run on any current OSes. It will not run on Windows 7, Server 2008, or anything since. So it stopped at 2000 and XP, that lineage.

Leo: I feel like it was written inside the Pentagon.

Steve: It was written wherever NSA is; you know.

Leo: Well, yeah, okay.

Steve: In Virginia, probably.

Leo: Virginia at McLean.

Steve: Yeah.

Leo: Yeah. Very cool stuff. Maybe Goodwill Hunting [crosstalk].

Steve: Oh, but to just - to look like it's doing complex calculations correctly and just like slip a digit.

Leo: So subtle.

Steve: Yes.

Leo: This is like, you nailed it, the Enigma project of the Brits, or "The Man Who Never Was." You know, the idea of just these subtle little...

Steve: And they never, never admitted to it, never said it. It never became public.

Leo: No one even knew.

Steve: No one ever knew that this thing was ever there before.

Leo: There's an Iranian nuclear scientist somewhere going, oh, now I understand why it didn't work.

Steve: Exactly.

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/>