Patch Tuesday la AI

Description: Rootkits found in more than 400 Arch Linux User Repository packages. The U.S. government requests Anthropic to remove Mythos and Fable. CISA responds to AI-driven attacks with new patching requirements. NPM to switch to more secure install defaults. Will it help? Our listeners react to last week's PHP commentary. June shows that AI has arrived for vulnerability discovery.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-1083.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-1083-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here, and he's raring to go. There are more supply chain attacks to talk about, including on the Arch Linux user repository, NPM. They're going to try to make that a little bit more secure. Steve has some recommendations until they do. And June shows that AI has arrived for vulnerability discovery. We'll talk about that and a whole lot more next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 1083, recorded Tuesday, June 16th, 2026: Patch Tuesday la AI.

It's time for Security Now!, the show where we cover the latest in security, privacy, and all that stuff online with the man, the myth, the legend. Not the Mythos. Mr. Steve Gibson.

Steve Gibson: No Mythos for you. That's true.

Leo: No Fable here.

Steve: No Fable, either. Wow. Crazy.

Leo: Wow, what a story, yeah.

Steve: Yeah. I'm going to share Anthropic's response, and then we'll do a little editorializing around that because that was an interesting weirdness.

So last Tuesday was June's Patch Tuesday. And, boy, did we break records. So today's podcast is titled Patch Tuesday la AI because, you know, this is what we were expecting

to see. It'll be interesting to see how long this tsunami, crescendo, tidal wave, pulse lasts. I don't expect it to be, like this is not going to be an every month thing. But five, six months would be my guess. And then we're going to see a reaction drop in the number of monthly patches because the AI is going to get deployed, in the case of Microsoft and Patch Tuesday, horribly named code name MDASH, which they make me say every time, will be the thing that, you know, changes, I really believe changes the Windows side of the industry. Anyway, we're going to dig deep into that. I want to talk about rootkits having been found in more than 400 Arch Linux user repository packages.

Leo: I have to really worry about that. I'm an Arch Linux user, yeah, mm-hmm.

Steve: Yeah. The U.S. government requests Anthropic, as we said, to remove both access to both Mythos and Fable for nationals. But since you can't, I mean, it's like the age restriction problem; right? It's like, well, we're not really sure so we just have to tell everybody no.

CISA has also had an interesting response to AI-driven attacks, changing their patching requirements for federal agencies. And boy, if patching was ever a backroom or like on the back burner, this is - it is just no longer the case. Any federal agency who - this is a BOD, I can't remember what that stands for, Binding Operational Directive from CISA which, you know, has legal strength which says you have to patch on the timeline we say. And it's fast. So patching, again, it's like this has moved right up to the front of, you know, operational readiness, business-wise. We'll look at that.

Also NPM, the most attacked repository we have, you know, the node.js packet manager, has switched to more secure install defaults. The problem is, a lot of non-malicious use of these install defaults occurs. And so this is going to create breakage which is going to have an unclear effect on long-term security. I found a really interesting analysis of this from somebody on the inside who understands what's going to happen that we're going to take a look at. Also a bunch of people responded across the spectrum about my little rant about PHP, which of course is not the first time I've ranted about PHP. But, you know, we've got great loop closure now with email communications. And so I'm going to share a bunch of that.

And then we're going to look at the consequence of AI having been here long enough, code-cognizant AI, long enough to have a serious impact on June's patches. So I think a lot of fun stuff to talk about and, of course, a Picture of the Week that...

Leo: Of course.

Steve: ...actually has a kind of a coding theme, or that's how I'm going to spin it, anyway.

Leo: Oh. Well, you know I'm always up for that.

Steve: Absolutely.

Leo: I have a little tale to tell about AI saving my bacon yesterday, too, at some point.

Steve: Oh.

Leo: Yeah. It was like a life saver, speaking of Linux. But we won't...

Steve: Oh. In that case, let's do the first sponsor. Then you have to tell us because I can't wait.

Leo: Okay, yeah, yeah. I just, the more I use it, the more valuable it becomes. That's what's really clear.

Steve: [Crosstalk]

Leo: Yeah. I miss Fable, but, you know, this was with 4.8, Opus 4.8. It did a great job. Now back to Steve and our Picture of the Week.

Steve: First tell us about AI saving your bacon.

Leo: Oh. So I run my agent, everything, all of it, all the models, all the memory, everything on that nice Framework desktop I bought, really designed to do that. I bought it to do local AI, 128GB of RAM. It's got that AMD AI+ 395, AI specific processor. It's really nice. Yesterday I come in, I mean, and I use it every day. That's where Claude is, where everything is. And it says "Emergency. Can't boot." And it gives me a prompt. And I go, oh, that's not good. Now, in the past what I would have done is I would have reinstalled Linux on that machine, and then had to go through - I've got backups of course, but I'd go through the hairy process of restoring it. And because it's my agent, I wouldn't have any assistance doing it. Fortunately...

Steve: Right.

Leo: Yeah. It would be on my own, buddy. Fortunately, I do have Claude Code running on my laptop. So I thought, well, let me just see if Claude Code can help. It said, "Don't erase it. Tell me what the message is." And I told it. It said, "Oh, good news, that might just be a minor thing. Let's figure it out." It had me type in - it started to have me type in a lot of stuff. Basically I went to - rebooted it through the Linux install USB key so that I have an operating system, and I can look at the hard drives. And I said, "This is a lot to type in. I'm going to make mistakes." It said, "Oh, oh, of course." This is Claude talking. "What am I thinking?" It said, "Look. I will write a..."

Steve: You're human.

Leo: You human. You poor human. "I need to SSH in to this machine. But of course you're running the boot thing. It doesn't know who I am. It's not let me ask this agent. So here's what I'm going to do. I'm going to write a little web server, run a little web server on the laptop here. And you're going to cURL the key, the SSH key,

into the boot thing, turn on SSH, and then I will SSH in and do it for you." I did. It did. It SSH'd in. It said - it looked and looked and said, oh, you know what the problem is? It's a little weird. But you have two - I don't even know what an ESP is. But you have two ESPs. The way I set this up was it's just dual SSDs, they're LUKS encrypted, and they're mirrored so that if one dies - and this is how mission critical - if one dies, the other one has still got everything on it. And it does, because the LUKS encryption, it has the password and the TPM.

So the other thing is it has to reboot if I'm not around. When the power goes out, I have to reboot, but I do want it encrypted, I just don't want to have to be there to enter the password. So it does that automatically to the TPM. I don't know what an ESP is, but apparently it has something to do with UEFI. There are two different ESP modules. And for some reason they had different Linux kernels. One had 7012, one had 7011. And that's what was stopping it from booting. It figured that out. It said, "Okay, I'll just - no problem. I'll just put the other kernel - I'll make them match. And from now - and I'm going to write a little stub now. When you do an upgrade, it'll automatically make sure it's the same kernel on both ESP modules so this doesn't happen again." It said, "Okay, all done. Reboot." It's fixed.

Steve: We're living in a science fiction world.

Leo: I would never - I don't even know what ESPs are. I would not have been able to fix this. It has to do with the UEFI boot. I wouldn't have known where to start. It would have been a massive chore. Instead, it wrote a little web server. I cURLed the key over. It logged in and did it all for me. Yeah. And you tell me these things are just autocorrect. Wow.

Steve: No.

Leo: It's very good.

Steve: It's funny, I had an experience yesterday. I'm configuring the house's security and environmental monitoring and management stuff.

Leo: Oh, it's really good at that, actually, yeah.

Steve: And I was using Claude. But I was - I had it turned down to Sonnet something. And it was kind of stumbling along, and I was getting some bad answers. And I sort of thought, what am I doing? Why am I using Dummo?

Leo: Sonnet's pretty good, but not for the hardcore stuff, yeah.

Steve: So I switched to Opus 4.8.

Leo: Yeah.

Steve: And what was interesting was that it's that model, in reviewing our dialogue, started to apologize about the previous bad answers that I had been given. It said, uh...

Leo: Yes, 4.8 is very apologetic.

Steve: I'm not sure what happened here, but, you know, I got - they were like, no, that was somebody else. That was just - that was Dummo.

Leo: That was Sonnet. Don't take the blame. That's one thing that 4.8 is notable for. It does apologize a lot. I don't - but it makes - but this one was so careful. It said I know this is, you know, I understand. This is your agent. This is mission-critical. I'm going to move slowly and make sure that I'm doing everything right. And it was flawless. It was...

Steve: Oh, and in fact I had to say, now that I was on 4.8, every answer was correct, and at one point it said, "Now, I'm going to make sure that I've not given you the wrong information by checking on [blah blah blah blah]." And I went off and did something. Then it came back, and it's like, okay, you know, I verified that this is the case. And it's like, and I'm just sitting here thinking, wow. I have an assistant.

Leo: I know some people listening are thinking, oh, Leo and Steve have lost their marbles. They drank the Kool-Aid.

Steve: Yeah, where is the Kool-Aid?

Leo: But until you've actually had this experience, it's easy to pooh-pooh it.

Steve: Yes.

Leo: But once you've had the experience, it's like, wow.

Steve: And as I said to our listeners a couple weeks ago, get an account. It can be free.

Leo: That's fine.

Steve: But that creates context. And my feeling is, if what we're saying sounds wacky, then it's because you haven't tried it. Or there's nothing you need it to do. The idea is, you know, if you're trying to do something, then - and also old habits die hard. I mean, this is all new. And so, you know, it takes a while to get used to the idea that there is this astonishing assistance available.

Leo: It's kind of amazing. Well, and the thing is, it speaks computer really well. It's very good at speaking computer.

Steve: It's funny, too, because that's significant. I remember when Google searches only turned up the results I wanted because only computer geeks were on the Internet.

Leo: Right.

Steve: And so the Internet was very computer-centric.

Leo: It was much higher quality, yes.

Steve: Now you've got all this, you know, all this nonsense out there, and so you get, you know, search results aren't nearly as interesting for computer people because it's got, you know, everybody else. So it's nice that it's like...

Leo: Anyway, that's my tale. I was just blown away. It was just really surprising, yeah.

Steve: Okay.

Leo: Thank you. Thank you, Claude, for fixing my computer. Picture of the Week time.

Steve: So I gave this one the caption "A loop with the branch test at the end."

Leo: Okay. I'm ready to look at it. I haven't seen it.

Steve: A loop with the branch test at the end.

Leo: [Laughing] Okay. I like it. This is a - now, Steve's turned this into a coder joke. But it isn't really a coder joke.

Steve: Correct. Correct.

Leo: Or is it?

Steve: So the sign says - so this is some signage somewhere, like at a service window or something, and it says, "Back in 15." And then on the next line it says "Minutes. If not, read sign again." So...

Leo: Go to 10.

Steve: So, exactly. So what we have here is a loop with a delay statement in it. And it branches back to the top. Now, for those who don't code, there are different configurations for looping. You are able to say, like for example, while something, then do the following.

Leo: Right.

Steve: Or you can say, when you're finished doing something, you can perform a test and maybe do it again. So, and it turns out these are, like, subtly different constructs in computer science. It's one of the things, like, programmers who are just starting out learn is that there are cases where you'd never want to execute some code that could be executed multiple times, which is why we call it a loop, because you loop back. In which case you would test for that decision at the beginning of the loop before you'd even done it once. There are other times where you always want to do it once. And then you want to see, well, do I do it again? And so that's where you put the test at the bottom. So anyway, I gave this one a loop with the branch test at the end because, you know, back in 15 minutes. If not, read sign again. So, yeah.

Leo: That's very funny. I love it.

Steve: Yup. Okay. So Arch Linux User Repository, abbreviated AUR, seriously compromised. More than 400 instances of, well, seriously, not only in count but in what, 400 instances of Linux rootkit and infostealer malware were found. The infostealer targets credentials and access tokens. We're going to dig somewhat deeply into that here in a minute because we've never really talked about infostealer malware. We've referred to it, you know, it's like, oh, right, it's an infostealer. But we haven't, like, what info is stolen? So we're going to answer that.

Last Thursday, the site, which I like, great domain, "ioctl," which is the abbreviation for IO Control, you know, input/output control, ioctl.fail is the site.

Leo: Oh, what a good name. What a good name.

Steve: And I didn't know there's like a .fail top level domain.

Leo: I didn't either. I've got to get that one, yeah.

Steve: Yeah. They posted their analysis of this infiltration campaign, opening their report by explaining: "This report summarizes static reverse engineering of the Linux ELF malware sample named 'deps' - deps is the name of the malware - and static review of the recovered NPM package source associated with the incident. The sample and package were treated as malicious throughout handling. No dynamic execution of the ELF, NPM package, lifecycle scripts, or package code was performed. The binary is stripped and implemented with Rust-style async state machines. Function names in this report are analyst-assigned names based on decompiled behavior."

Okay. So to interrupt for a minute, what they're saying here is that at no point did they actually execute the malware under any context. Now, that's sometimes done; right? You run it in a protected virtual environment in a sandbox in order to watch it go, see what it

does. And just for the record, this "ELF," this E-L-F, which is referred to here and throughout for those not well versed in Linux lingo, is the abbreviation of Executable and Linkable Format, thus ELF, Executable and Linkable Format. It's a very flexible format that's used almost universally by Linux and the UNIXes also, and as well as some embedded RTOSes, and pretty much everywhere other than Windows, which, you know, has its own PE, the Portable Executable format, and Mac which uses Mach-O as its format.

In any event, they chose not to let this thing, as I said, loose, I don't know why, in a sandbox, you know, such as a secure VM. The idea there would be any damage that it might attempt to do could be observed and contained. Instead, they reverse-engineered the malware from a static binary sample of the malware code. And since this malicious binary had been stripped of any latent symbolic names, it was just pure, you know, binary code that did the work, you know, variable names or function names were not there, the analyst assigned the names as they worked through the decompilation of this once a function's purpose had become clear.

So they go on to explain: "This sample was recovered from a supply-chain compromise involving an Arch User Repository package build flow. In the reported intrusion path, the attacker modified AUR build steps so that the build process downloaded and installed a malicious NPM package. That package masqueraded as atomic-lockfile," which is like a useful thing, "version 1.4.2." So that itself is not malicious. It was masquerading, pretending to be atomic lockfile v1.4.2 and included the Linux ELF payload at src/hooks/deps is where it was in the file system.

"The malicious NPM package uses a preinstall lifecycle hook. Now, this is interesting. We're going to be talking a little about this whole thing coming up a little bit later. They preinstall lifecycle hook to execute the ELF automatically during NPM installation." Could be useful. In this case it's a source of abuse. "This means," they wrote, "a developer workstation, a maintainer's machine, or CI/ build host could execute the malware as a side effect of building or installing the compromised AUR package. 'Deps'" - that's the malware - "is a Linux credential stealer with optional root-only eBPF rootkit capabilities."

eBPF, I'll just interrupt, refers to the "extended Berkeley Packet Filter" technology that allows user-provided code to be run in the Linux Kernel. This gives that code direct access to the most privileged information in the system. You know, normally, I mean, this is very useful for analyzing communications. You need to be down in the kernel in order to reduce the analysis overhead, thus the extended Berkeley Packet Filter." This is an abuse of that privilege, essentially.

So they continue to describe the malware, writing: "It's designed for developer workstations and build environments. It targets browser and Electron application data, Slack, Microsoft Teams, Discord, GitHub, NPM, Vault, Docker/Podman, SSH, VPN material, shell histories, and other local developer secrets." Yikes! In other words, no developer wants this thing anywhere near their machine.

They said: "The recovered supply-chain package identifies itself as atomic-lockfile version 1.4.2. It contains a malicious NPM lifecycle entry: 'preinstall' and './src/hooks/deps.' That lifecycle script executes the ELF directly during NPM installation when lifecycle scripts are enabled. The ELF in the package source is byte-identical to the analyzed sample." So that was a little bit of repetition, and they're just saying what they analyzed was byte-identical.

They said: "The attacker-controlled," you know, "command-and-control (C2) endpoint was recovered from the ELF. It's not supplied by the NPM package, command-line arguments, or a JavaScript wrapper. The binary decodes an onion service address at

runtime." And then they give the address. It's just a long string of gobbledygook like all the onion addresses are, ending in .onion for the domain name.

"The command/result callback is a POST /api/agent, sent through a local loopback/SOCKS-style transport. The local 127.0.0.1 traffic is an intermediate transport layer, not the attacker endpoint. When eBPF is available, the malware can hide local process and socket artifacts used by the transport."

Okay. So in other words, if the code has access to the extended Berkeley Filter Functionality, then that access will be used in a rootkit-like manner to completely obscure any and all evidence of any infection for the software itself. It just disappears so that anyone looking at the machine, looking at, you know, the equivalent of netstat, looking at whose, what processes are listening for connections on ports, what communications are occurring on the fly, you know, looking, for example, for a connection to a command-and-control server, it just won't show up. The rootkit system that it brings completely makes it just vanish. And we of course covered rootkits way back in that Sony infiltration, the Advanced Persistent Threat that got Sony. So they said: "Without eBPF, the local network presence of 'something' could be observed.

"So the recovered package," they wrote, "the recovered package source appears to be a mostly legitimate TypeScript NPM package with a malicious ELF inserted into the source tree and then wired into that NPM lifecycle execution." In other words, it's a normal benign piece of big JavaScript, you know, TypeScript. And they just sort of tacked in this ELF binary and then used the NPM preinstall functionality to get it, you know, to execute it, and then it would do all of its bad stuff.

They said: "Static review of the package source outside the ELF found no JavaScript wrapper, no additional command-and-control configuration, no command-line arguments passed to the ELF, and no package-layer references to temp.sh, /api/agent, Discord webhooks, or a public C2 domain/IP." In other words, the Typescript NPM package was clean, did not contain any other infection. The infection, such as it was, was all in its ELF binary. So they said: "Conclusion: The malicious NPM package provides the execution vector. The C2 endpoint is encoded inside the ELF itself."

Okay. So now we know what it is and how it's delivered and carried into the system. Which brings us to: "What does it do inside the developer's machine once it takes hold?" And remember, there are 400 instances of these bad things that were discovered in the Arch Linux repository, the user repository. So this no doubt infected a bunch of people. So we learn further why no developer wants to have this anywhere near the system.

So it installs persistence using root or per-user systemd service units; enforces a single active instance using flock(), so to keep like multiple instances from running at once; redirects standard input/output/error to /dev/null, so that you won't see it doing anything, any of its output; ignores SIGPIPE, so it takes itself out of any external control; reads /proc/self/exe to locate and copy and install its current executable; uses Rust async runtime logic to run collectors and transport tasks; enumerates Chromium-family browser profiles and Electron app data; reads SQLite cookie databases and LevelDB local storage; extracts Chromium and Electron cookies and service tokens; Queries Slack, Microsoft Teams, Discord, GitHub, NPM, and OpenAI/ChatGPT APIs with stolen tokens or cookies.

Searches local filesystem locations for SSH keys, your shell history, Vault tokens, Docker/Podman credentials, VPN material, and developer secrets; uploads file content to temp.sh; calls back to the recovered onion command-and-control over a POST query to /api/agent; uses a local loopback/SOCKS-style transport before reaching proxied destinations; includes a downloader/stager path tied to /usr/bin/monero-wallet-gui. And if sufficiently privileged, loads an embedded eBPF rootkit to hide its processes, its process

names, and all of its socket nodes. So it goes completely stealth once it gets into your system if it's able to use eBPF. And again, just to reiterate, more than 400 instances of Arch Linux Repository packages have been found infected with this nastiness.

So as I said before, we've referred tangentially to infostealer malware from time to time. It is unfortunately an increasingly prevalent form of malware because its goal is obtaining information that would allow an attacker to pivot to some other target. They don't really care about a developer's machine. You know, but they're hoping that they'll get into the machine of a developer who also happens to have, for example, AWS credentials for some other juicy target, like the company he works for, or consults to, or something. So the bad guys are much less interested in the initial victim in this case than in what other systems or networks that victim may have access to. And of course the classic example was the LastPass developer who had a bug in his NAS software that was way out of date, and the bad guys got into him, found out that he was a developer at LastPass, and then got into LastPass.

So we've never looked closely at infostealers. Since they are definitely something that no one wants to discover in their systems, since they are growing in prevalence, and since we have a very nicely reverse-engineered infostealer here to take a look at, I want to share the details of what info this representative infostealer steals.

"The malware targets developer and collaboration data. And they enumerate them. This is literally what they found this code doing. It digs into the browsers and Chromium Profile Stores for Google Chrome, Chrome Beta, Chrome Dev, Microsoft Edge, Edge Beta, Edge Dev, Brave, Brave Beta, Brave Nightly, Vivaldi, Opera, Opera Beta, Opera Developer, Yandex Browser, Epic Privacy Browser, Iridium, Ungoogled Chrome, Thorium, Comodo Dragon, SRWare Iron, Cent Browser, Slimjet, Maxthon, UC Browser, Coc Coc, Naver Whale, Chromium Flatpak, Google Chrome Flatpak, Microsoft Edge Flatpak, Brave Flatpak, Vivaldi Flatpak, Opera Flatpak, and Yandex Browser Flatpak." In other words...

Leo: I never heard of half of those.

Steve: I know. But they exist. And so these developers took the time to put in some specific code for each and every one of those because, if a developer has it, they want to get into it.

"Profile artifacts targeted include: Local Storage/leveldb, Network/Cookies, Cookies, Default/Cookies and Chromium encrypted cookie values. "Collaboration and Electron Applications," which it knows about and goes after: "Slack, Slack Flatpak, Slack Snap, Microsoft Teams, Microsoft Teams legacy stores, Microsoft Teams Flatpak/browser-derived stores, Discord, Discord PTB, Discord Canary, Discord Flatpak variants, Discord Snap variants, Vesktop, Legcord, WebCord, ArmCord, Vencord, NativeCord, Abaddon, Dissent, Ripcord, and Datcord." They also "Confirmed Slack paths and data."

Leo: Wait a minute. Ripcord and Datcord?

Steve: That's right. They also confirmed Slack paths and data where this thing looks at it, ".config/Slack, .var/app/com.slack.Slack/config/ Slack, snap/slack/current/.config/Slack, Slack d cookies..."

Leo: If you just tuned in...

Steve: "...for *.slack.com, Slack API enrichment through /api/auth.test, /api/users.info, and /api/conversations.list." And we're about halfway through.

"Confirmed Microsoft Teams and Microsoft service artifacts include: .config/Microsoft/Microsoft Teams, authsvc.teams.microsoft.com, teams.microsoft.com, skypeToken, regionGtm, cache.token, Authorization: Bearer, X-Skypetoken, Teams account, tenant, and team metadata."

They've confirmed Discord artifacts, which it digs around through, including Discord tokens from Electron/browser storage, /api/v9/users/@me, /api/v9/users/@me/guilds, MFA state, premium/Nitro type, flags, guild ownership, permissions, and member-count metadata. Developer Accounts and Package Ecosystems including GitHub, NPM, and OpenAI/ChatGPT account metadata.

"Confirmed GitHub strings and endpoints include: api.github.com, where they get a bearer token, user-agent. They look for credentials, account, and repository metadata such as login, company, public repository count, followers, and repository stars. NPM strings and endpoints including .registry.NPMjs.org. They get the package publishing identity and maintainer package metadata.

"The OpenAI/ChatGPT path queries api.openai.com with stolen bearer material for account metadata. This is - so it's credential validation/enrichment against a third-party service" and, they note "not evidence that OpenAI is attacker-controlled infrastructure."

And finally, "Local Developer Secrets: Vault token files, Docker command history and registry credential material, Podman command history and registry credential material, SSH keys and SSH configuration, PuTTY private-key material, VPN profiles and .ovpn" - that's OpenVPN - "files, shell histories for bash, zsh, and fish, command history containing sftp, ssh-keygen, ssh-copy-id, ssh-add, rsync, putty, plink, docker, docker-compose, and podman commands." And yes, I know that was a lot.

Leo: I've made a song out of it.

Steve: Oh, god.

Leo: Would you like to hear the...

Steve: Oh, we have to have it.

CLIP: ...Microsoft Teams, Microsoft Teams legacy stores, Microsoft Teams Flatpak/browser-derived stores, Discord, Discord PTB, Discord Canary, Discord Flatpak variants...

Leo: It's a long song. You don't want to hear...

Steve: It's really pretty good.

Leo: Here's a little more.

CLIP: ...Discord Snap variants, Vesktop, Legcord, WebCord, ArmCord, Vencord, NativeCord, Abaddon, Dissent, Ripcord, and Datcord...

Leo: I'm sorry. I'm sorry. Go ahead.

Steve: It's wonderful.

Leo: It's amazing what AI can do. Actually we shouldn't laugh because this has been a nightmare for everybody using Arch. I mean, it is the worst thing you could possibly have happen. Terrible. Terrible.

Steve: Yes. So as I said, I know that what I just did to everybody was a lot. But I think it's important to appreciate that the more than 400 instances of this malware which were discovered residing in the Arch Linux Repository were expressly designed to root out any and all instances [clip returns].

Leo: Sorry.

Steve: ...any and all instances of any of that developer data and send it back to its command-and-control server. So the real takeaway here is this infostealer stuff, this is what an infostealer looks like. It's what it feels like. It's what it does if it gets into your computer. It's out there, and developers really need to be extremely cautious, more so than ever, that this doesn't get into their system because it's going to elevate its privileges. It's going to rummage around through your system, sing a little jingle to itself, and just suck everything out of your computer that you just take for granted. And the developers will get it and use it against you, or anybody whose information you have on their behalf - AWS, an employer, someone that you're consulting for, and so on. So it's real, and it's bad.

Okay. So last Friday afternoon, at the request of the U.S. government's nonspecified concerns for national security, Anthropic shut down all access to their two most advanced models: Claude Fable 5 and Mythos 5. And I'm sure everyone has seen this a lot. We should note, though, that claiming "national security" has become the catch-all phrase used by the U.S. government, which should be taken to mean either "because it's what we want" or "because we say so." So it's, you know, often not very satisfying. It's not at all clear why this is the case. But in any event, since it was just, Leo, it was at the start, the very start of last week's podcast that Claude Fable 5 first appeared.

Leo: I think I mentioned it, didn't I, on the show?

Steve: Yeah, you did. It's like, hey, there's a new model. And in fact you began playing with it. I think it was during the podcast.

Leo: Instantly, yeah.

Steve: You ran it on some of your existing code.

Leo: Oh, that's right.

Steve: And it found a whole bunch of more stuff.

Leo: It did. It found a lot of security flaws. It was very good.

Steve: So it looked like another major leap forward. So as a consequence of all of that, I want to share what Anthropic posted because there's some interesting pieces of sort of things to read in between the lines here about why their two newest models have been taken down.

And as I'm saying this, listen to the language Anthropic uses when they talk about safeguards and jailbreaks because this is their pushback. And it echoes the position everyone here has heard me articulate from the start, which is that our - to me it feels, understanding enough of how this large language model technology operates, intuitively it feels as if it is - the whole technology is almost certainly going to be inherently hostile to any form of control. And I don't mean hostile like in a belligerent way, but I mean it just - this isn't something that can be controlled. The way it works is not like normal procedural code. It's not the way it is. You know, you've got temperature that you can turn up and down. So I believe it is inherently an uncontrollable technology, and that is from the standpoint of guardrails.

So the headline of Friday's posting - and as you'll see, they talk about that. The headline of Friday's posting was "Statement on the U.S. government directive to suspend access to Fable 5 and Mythos 5." So their statement is: "The U.S. government, citing national security authorities, has issued an export control directive to suspend all access to Fable 5 and Mythos 5 by any foreign national, whether inside or outside the United States, including foreign national Anthropic employees. The net effect of this order is that we must abruptly disable Fable 5 and Mythos 5 for all our customers to ensure compliance." It hadn't occurred to me, Leo, but I guess some of their employees have to be, like, denied access somehow. Which is wacky. "Access to all other Anthropic models will not be affected," they said.

"We received the directive from the government today at 5:21 p.m. (ET). The letter did not provide specific details of its national security concern. Our understanding is that the government believes it's become aware of a method of bypassing" - or "jailbreaking" they have in quotes - "Fable 5." And I saw elsewhere, Leo, that apparently some Amazon employees or someone at Amazon, like, believed they figured out a specific jailbreak? I think that was the case. Anyway, they said: "We reviewed a demonstration of this specific technique being used to identify a small number of previously known minor vulnerabilities. These vulnerabilities all appear relatively simple, and we have found that other publicly available models are able to discover them, as well, without requiring a bypass.

"Anthropic's posture with respect to Fable's safeguards, as laid out in our launch blog post, is the following." So they said, and we have some bullet points: "We have instituted strong safeguards that greatly reduce the likelihood that Fable is misused for tasks related to cybersecurity, among others." Remember, as we know, they just put a blanket lock on cybersecurity and bio something or other, saying sorry, Fable, just don't answer any questions about that. And as we saw, they, like, they sort of kick you out to the Opus family if you make the mistake of crossing one of their tripwires.

They said: "In fact, our safeguards are so strong that many users have complained that they are overly broad." Right. Again, because that seems like the only way to do this, if you're going to try to do it, is to fail closed because it just isn't possible to control these models. So this is an artifact of that lack of ability to control is they have to do just blanket, like, overreaction.

They said: "In the weeks leading up to the launch of Fable, Anthropic worked with the U.S. government, the UK AISI, multiple private third-party organizations, and internal teams to red-team Fable's safeguards for thousands of hours in total. These tests showed that Fable's safeguards are substantially more effective than those of any previously deployed model.

"No testers have yet been able to find a universal jailbreak" - we'll see why that word's important in a minute - "a universal jailbreak - a jailbreak method that can very broadly bypass the model's safeguards, unblocking a wide range of cyber capabilities.

"We suspect that perfect jailbreak resistance is not currently possible for any model provider." Think about that. "We suspect that perfect jailbreak resistance is not currently possible for any model provider. Every safeguard used in the industry is vulnerable to non-universal jailbreaks, which can elicit some cyber information in specific circumstances. And it is likely that universal jailbreaks will eventually be found in the future. We stated this clearly when we released Fable 5.

"Given that perfect jailbreak resistance does not appear to be possible today, Anthropic adopted a defense-in-depth strategy with Fable 5. We aimed to make jailbreaks either narrow (in the case of non-universal jailbreaks) or very expensive to produce (in the case of a universal jailbreak), and to combine this with thorough monitoring to quickly detect and shut down any successful attacks. This is also why Anthropic has required 30-day retention of customer data with Fable, a policy change that carries real costs for us with customers, but that allows us to research and mitigate jailbreaks." So like, you know, by holding onto 30 days' worth of interaction, if they find that Fable has been subverted, then they're able to go back in time. There's nothing that the bad guys can do to erase that history that allows Anthropic to then understand what happened and improve the jailbreak technology.

They said: "We stand by this defense in depth strategy. It reduces the risks posed by Fable, making them comparable to the risks of existing models already deployed across the industry."

Leo: And that is important, that line is really important, yes.

Steve: Yes. In other words, those that have not been banned by the U.S. government because the U.S. government says, yeah, these are fine. So this is - they believe Fable is as safe as any of their other models.

They said: "We have not even received a disclosure of a concerning non-universal potential jailbreak that led to a harmful result. The potential jailbreaks that have been disclosed to us are either entirely benign responses or are minor findings that provide no Mythos-specific uplift. To date, the government has only given us verbal evidence of a potential narrow, non-universal jailbreak, which essentially consists of asking the model to read a specific codebase and fix any software flaws. Our understanding is that one potential jailbreak was shared with the government. We've reviewed a report that we believe is the basis of the government's directive and validated that the level of capability displayed here is widely available from other models (including OpenAI's GPT-5.5),

again." And they said: "And is used every day by the defenders who keep systems safe. We'll share more details over the next 24 hours."

They said: "We are complying with the government's legal directive and are removing access to Fable 5 and Mythos 5 for everyone. However, we disagree that the finding of a narrow potential jailbreak should be cause for recalling a commercial model deployed to hundreds of millions of people. If this standard was applied across the industry, we believe it would essentially halt all new model deployments for all frontier model providers.

"As we have stated publicly, we believe the government should have the ability to block unsafe deployments, as part of a statutory process that's transparent, fair, clear, and grounded in technical facts. This action does not adhere to those principles. We apologize for this disruption to our customers. We believe this is a misunderstanding and are working to restore access as soon as possible."

Well, that was Friday. And Saturday, Sunday, Monday. And when I was using Claude this morning over the little prompting area...

Leo: Oh, it's still blocked.

Steve: ...it said Fable is currently disabled.

Leo: Increasingly this looks like a political move, a business move. Pete Hegseth...

Steve: Well, we know that Hegseth is pissed off at Anthropic.

Leo: He tweeted, "You see?" You know? I mean, he clearly - so Katie Moussouris, who is a well-known security researcher...

Steve: Yup.

Leo: ...was apparently given the paper. We think it was Andy Jassy, the CEO of Amazon, who talked to the White House and said, yeah, there is a jailbreak. This is the third-party research paper. Anthropic shared it with Katie Moussouris. She said, "As far as I know, I am the only person who's seen this." This is the - you want to know the jailbreak? Three words. Fix this code. The researchers took open source code with known CVEs plus new code with deliberately planted vulnerabilities, and asked Fable 5 to review the code for security issues. Fable 5 refused. They then asked the models - by the way, they also did Mythos and Opus - to fix this code, and through a multistep and manual process turned the output into scripts that test the patches. This is the jailbreak. By the way, Katie has made a shirt to kind of play on the previous time this commerce department blocked [crosstalk] munitions is crypto.

Steve: Web crypto.

Leo: Remember they made a T-shirt with the crypto on it - and you can't block a T-shirt, First Amendment protects it - and went across borders with it. She's done

the same with this. She's made a Fix This Code T-shirt, and on the back it says "This shirt is a munition." Now, this is her assertion. I don't, you know, it's possible there are other jailbreaks. I had heard that Pliny the Liberator, who's a well-known jailbreaker, had another jailbreak. We're trying to get Pliny on the show tomorrow. Haven't had any success yet. But we do have Alex Stamos joining us tomorrow. Alex, as you know, really well-known security researcher, has released a letter to Secretary Lutnick and National Cyber Director Cairncross asking them to free Fable, to lift Fable's restrictions. It's a very well-thought-out, explained letter. And it's signed by a great many security gurus including Alex, who wrote the letter.

Steve: And it's annoying that this comes from a competitor; right? I mean, that's what really feels wrong.

Leo: The Andy Jassy, well, the funny thing is Amazon has a huge investment in Anthropic. So I don't know if they're a competitor. This is them - I don't know, we don't know what Andy Jassy told Lutnick. But we do know that Anthropic has been very good to the administration. Anthropic has...

Steve: And is being used throughout the administration. I mean...

Leo: And is being used. Yeah. We also know that OpenAI has donated to the administration. I think it's very hard to - and maybe - we don't know what's going on. But there's never been provided any good evidence that this a real jailbreak. If it's Fix This Code, that's absurd. That's absurd. So Ed Felten is a signatory on this, many names that our audience would know very well.

Steve: Well, I think that the strongest case that Anthropic made in their response, to my mind, was this isn't that different from everything else.

Leo: Exactly.

Steve: You know, you've singled us out.

Leo: You could do the same thing.

Steve: You've singled us out. And, you know, we could suggest maybe this is the flipside of, like, of the consequence of all that marketing attention which many have called hype that Anthropic deliberately created to accompany that restricted access to Claude Mythos Preview. So, I mean, like they painted a target on their back. But again, you know, I think their strongest point is, hey, yeah, we're proud of this model. We're making it better. We've put in really strong safeguards. And we're not that different from any of the others.

Leo: It's a wakeup call for foreign national researchers and all of the AI companies in the United States. You've got to think they're thinking about what their plans are going forward, maybe leave the United States. It's certainly a wakeup call for the EU if the U.S. can block useful AI from the EU.

Steve: Wow.

Leo: They're going to be working on their models. It's I think a very big tactical error. And I think it's more about politics, pique, and the corruption than it is about actual security. But it's hard to say because we haven't seen - the only jailbreak we've seen clearly is not a problem. Maybe there are other jailbreaks. I understand why the government might say, you know, this AI model is too dangerous to be released. But they need to [crosstalk].

Steve: Well, and, you know, the other thing we've seen is other people achieving Mythos-grade success with non-Mythos model. So, you know, and we've talked about that here. It's not as if there's like some really magic pixie dust that Anthropic uniquely has. They just, you know, they're pushing ahead, as is everybody else. It's a good model, yeah.

Leo: It's a good model. We will talk to Alex Stamos tomorrow on Intelligent Machines, 2:00 p.m. Pacific, 5:00 p.m. Eastern.

Steve: Good.

Leo: Hope everybody will join us for that, and he will talk about this open letter which he has put out at freefable.org.

Steve: Free Fable, that's great.

Leo: Free Fable.

Steve: Okay. As we saw last week, the change to the historical attack pattern that was already accelerating, prior to the rise of AI, you know, I mean, before this all happened we were noticing, wow, like this is an accelerating problem. I made the comment, you know, a couple years ago about how quaint it was that we used to have three- or four-digit CVE numbers. And now they're all five-digit, and they're high five-digits. So will access to significantly more advanced autonomous AI-attacking capabilities by a much wider range of attacker change this further? And it absolutely clearly seems that we are seeing an AI-driven acceleration. And it will, for the first time, unfortunately aided by AI, include those who are far less knowledgeable and skilled. So it broadens the base of attackers because they don't have to be experts any longer.

Just as we've seen many of this podcast's non-coders delighting in their newfound ability to use AI to code, it's only a matter of time, and probably not much time, until we see attacks managed by non-hackers using next-generation AI-driven autonomous attack frameworks. Last week's report from Anthropic's red team, which is what we shared, showed one instance of exactly this that was, by the end of March, still the exception to the rule. They said, you know, not everybody else is using AI in post-intrusion strategizing, but that one group did. So a year from now it's going to be commonplace.

So against this backdrop, last Wednesday on June 10th, CISA released their "Binding Operational Directive (BOD) 26-04," titled "Prioritizing Security Updates Based on Risk."

This BOD formalizes - this is me speaking - formalizes the timeline under which federally mandated IT systems are now required to respond to cyberthreats that CISA identifies. We've already seen and commented about the response speed required by CISA to some recent threats. Remember there was one like by midnight that night, or within three days. But now that somewhat breathtaking three-day-to-patch timeline has been formalized.

I've got a chart in the show notes at the top of page 8 which actually shows the decision tree that you follow in order to determine how quickly you must patch something. The guidance that CISA has produced has been reduced to this tree with four binary decision points. Since 2 to the power of 4 is 16, that is to say, there are 16 combinations of four bits, we've got 16 leaves at the end of this decision tree.

Leo: This is worse than the FIFA chart here. This is - holy cow.

Steve: The four branching decisions are: Has the vulnerability been publicly released? Is the vulnerability in KEV, the Known Exploited Vulnerabilities list? Is the exploitation of the vulnerability automatable by the adversary? And is the impact of its exploitation partial or total control? Each one of the 16 branches ends in either three days to patch - that is to say, you have three days to patch this, guys - or 14 days to patch, or 60 days to patch, or patch on system upgrade. Meaning, eh, don't worry about it. When you upgrade everything else, you know, you'll get patches. So it's essentially patch it immediately, like now, with a deadline in three days; or you've got two weeks to patch this; or you've got two months to patch this; or don't worry about it that much.

So, for example, riding along the worst-case branches of the tree for each decision point, a known vulnerability, a publicly known vulnerability that's on the KEV list, which can be automated and carries an impact of "total control" by the attacker, that guy must be fixed in three days. Like do it now. But the deadline is three days away. Now, the flipside, following the best-case branches, if the vulnerability has not been publicly exposed, is not on the KEV list, cannot be automated, and regardless of whether or not its impact would be partial or total on the victim's system, turns out no hurry, can be fixed during routine system updates. But all other combinations of those four criteria bring you out to a leaf on the tree that tells you how much time you have.

What this means is that all federal agencies falling within CISA's Binding Operational Directives will need to put a system in place, I mean, again, it's no longer like, oh, yeah, patching. George does that, but he's on vacation. No. Patching has come front and center. Maintaining up-to-date systems is no longer something that government agencies will be able to give lip service to while planning to get around to it, you know, whenever.

So those lazy days are over, thanks to AI. And I'd be pretty certain that they're never coming back since once those systems have been painfully put in place, like rapid patch cycle ability, why would they not continue? So, you know, it took clear and present threat from AI to push the change which nobody wanted, I mean, this is going to upset a lot of agencies who said, oh, we have no ability to actually do that. Well, they're going to have to figure it out and generate that ability.

Leo: I think that's a good thing, though; right?

Steve: It is a good thing, yes. I 100% agree, Leo. I think that it's not anything anyone wanted...

Leo: You've got to patch.

Steve: But patch, you've got to patch.

Leo: Yeah. By the way, I hope this is okay with you for the album art for the show. The Free Fable march.

Steve: [Crosstalk]; isn't it.

Leo: I know, I know.

Steve: That really looks like you. Some of these things. And that really looks like me, actually.

Leo: Actually it looks like, if you didn't know better, you'd think we were in fact leading a march to free Fable.

Steve: Yeah. And I think these photos must have come from the show because, I mean, it doesn't look AI.

Leo: It's Darren. He's very good. Darren is our most adept AI user, I think, at this point. He's quite good. Back to Steve and Leo, radicals. The radicals, marching to free Fable. Go ahead, Steve.

Steve: So the news is that the much-attacked NPM (Node.js Package Manager) will be flipping its defaults to begin disabling the auto-running of installation time scripts starting with the July 2026, so next month, release of its version 12.0. The change hopes to counter the massive rise that we've been seeing, we talk about it every week almost, in the number of supply-chain attacks taking place on that platform. As we've seen here, threat actors are increasingly hiding malicious commands inside install scripts that get auto-executed when a victim installs a new package. We were just talking about that with the Arch Linux. This sounds welcome and great; right? I mean, it's like, hey, it's automatic. It's nice. But what effect will it have, that is, flipping this from by default enable to default disable?

I tracked down an informed opinion of someone who should know, writing for the blog "opensourcemalware.com," to get a reality check. The posting was titled "NPM disables install scripts by default, but is that going to solve its malware problem?" And the blog's tagline was "NPM announced that the new version of the NPM package manager (v12) will come with several security improvements, including disabling install scripts. Is this a game-changer or security theatre?"

Its author wrote: "On June 9th, NPM announced the breaking changes coming in v12." And he said: "I'm genuinely excited about this announcement. Three permissive defaults that have shipped malware to developers for a decade are about to flip to deny-by-default. Pair that with install-time cooldowns" - I'll explain what that is in a second - "landing across every package manager, and the rise of commercial supply-chain

firewalls, and you'd be forgiven for thinking that NPM malware problem is finally getting solved.

"A year ago," he wrote, "I stood on a DEF CON stage and walked through why NPM install scripts are terrible. So let me be the first to say it: These coming v12 changes are good. They're also, in part, theater, and understanding which part is which is important. NPM v12 flips three dangerous defaults. The big one is 'install scripts.' Today, NPM install happily runs preinstall, install, and postinstall hooks from any package in your tree, including transitive dependencies you've never heard of. In v12, that stops by default - no automatic lifecycle scripts, no native node-gyp builds, no prepare scripts from git, file, or link dependencies. You opt packages back in with NPM approve-scripts, discover what's affected with NPM approve-scripts, the allow-scripts-pending option, and block the rest with NPM deny-scripts.

"The second change makes git dependencies require an explicit allow-git option, closing a code execution hole where a git dependency's own .npmrc could override the git executable, a bypass that worked even with the ignore-scripts option set. The third and final change makes remote URL (HTTPS tarball) dependencies require an allow-remote option. These three changes," he writes, "ship around July 2026, and you can prepare today in NPM 11.16.0 and subsequent.

"At the same time other security improvements are taking hold. 'Cooldowns' are also now everywhere. NPM shipped with support for 'min-release-age' in 11.10.0 back in February. This is a setting that refuses to install a version until it's been public for a configurable number of days."

Leo: Yes.

Steve: "The logic here" - isn't that nice? Yes.

Leo: I turn that on. Well, I do it manually, but for 14 days, if it's not more than 14 days old, I don't want to install it.

Steve: That's smart, yes. And he writes: "The logic is simple and effective: compromised releases usually get caught and pulled within hours, so a short delay filters most of them out at the install layer with zero scanning. It's worth noting," he writes, "that NPM was the last to this party. pNPM shipped minimumReleaseAge in 10.16 in September '25, yarn added npmMinimalAgeGate in 4.10.0 the same month, and bun followed in 1.3. There's even an open proposal to make seven days the default, which is the right instinct. Almost nobody needs a package the instant it's published.

"Supply-chain firewalls have become a product category. Developers are finally installing something on their machines that address malicious package threats. Datadog released an open-source supply-chain-firewall in 2024, and Liran Tal released his tool npq back in 2017. Tools like these wrap package managers and check to see if the user is about to install vulnerable or malicious packages; and, if so, they'll block them from being installed. There has been a lot of new tooling in this domain recently with Socket, Aikido, and Endor Labs all offering products in this space. 'Package Firewalls' like these work, but of course they rely on developers to not bypass their controls to install malicious packages anyway."

He says: "I don't want to be cynical about the right things. Killing automatic install scripts is the single most important change NPM could make. Install hooks are the mechanism

behind a huge share of the incidents we document at OpenSourceMalware. They're how a poisoned transitive dependency gets to run arbitrary code on a developer laptop or a CI runner the instant it's pulled. The security community and pNPM have been arguing for deny-by-default here for years. NPM finally agreeing is a genuinely good day. Cooldowns are the cheapest high-value control in the entire ecosystem. Firewalls and package managers give teams a real shot at stopping a supply chain attack before it lands. None of this is fake security. The problem is what happens next.

"The first observation is that 'off-by-default' only works if it stays off. The thing the announcement glosses over is that disabling install scripts is going to break a lot of stuff. An enormous amount of legitimate software gets installed, built, and wired into your application environments via install scripts." Right? That's where the magic comes from. That's where I'm - it's incredible. He said: "This, of course, explains why they're such a strong attack vector. There's a simplistic narrative going around that lifecycle scripts only exist to do bad things, and that's just not true.

"NPM packages are not just a way to import libraries. Many people, including me," he writes, "build CLI tools to do necessary utilitarian functions, and many of those tools use install scripts. Some examples of popular packages that use lifecycle scripts are: esbuild with 200 million weekly downloads; sharp at 60 million per week; core-js at 40 million per week; puppeteer, 10 million weekly; Nx, 9 million; bufferutil at 6 million; utf-8-validate at 8 million weekly downloads; and bcrypt at 5 million weekly downloads.

"Additionally," he finishes, "native modules need node-gyp to compile against your platform at install time. Tools that download a platform-specific binary, generate a config, register a shell completion, or build a native add-on all lean on install scripts to get the job done. This is not an edge case. It's a meaningful slice of the most-depended-on packages in the registry. The day version 12 lands, those packages stop working until someone approves them."

So I'm going to interrupt here for a second. If anyone listening is thinking to themselves, well, convenience versus security, you're exactly correct. That's what this is. I'm sure that many of us who have used package managers have been somewhat amazed - I know I have - by the astounding degree of automation. You fire it up, and all this stuff happens. Packages are grabbed from here and there. Compilations are run and linked together, packages are installed, everything just whizzes by on the console. What just happened? Who knows? Nobody knows. Well, someone knows, presumably. But the entire point is that we, the developer or the end user, who invoked the package manager, doesn't need to know. It all just magically happens.

And that is also, of course, the Achilles heel of the entire process. It's what opens it to such abuse because that extreme magical ease-of-use can be, and increasingly is being, abused to do malicious things behind our backs automatically. We have no idea what's happening in the first place, so how would we know if something bad was happening? The problem with flipping these magical defaults to off is that the magic, upon which we've become dependent, breaks.

So he continues his post, writing: "So what will developers actually do? They'll approve. Then they'll approve again. And by the third time a build breaks at 5:00 p.m. because some transitive dependency needs its postinstall hook run, the approval becomes a reflex. The deny-by-default protection quietly degrades into a click-through prompt." How would you like a cookie? Do you agree to have a cookie? We've all been there; right? Those darn notices all the time, it's like, yeah, okay, fine, yes, right. He says: "A click-through prompt that fires constantly trains people to click through. We've watched this movie with browser permission dialogs and OS security prompts. There's no reason to expect NPMs to end any differently. The control is real; the human standing in front of it is the same human who has a deadline.

"If you really want to know if disabling install scripts will have the intended effect, you can look over to the VS Code ecosystem. With VS Code before version 1.109, the global allowAutomaticTasks setting was on by default. This meant that malicious task files would automatically run if victims opened source code that included those task files. Microsoft changed and disabled this feature to be disabled by default in January '26, the beginning of this year, after months of threat actors used malicious tasks files to compromise developers. Did that stop threat actors from continuing to use VS Code tasks? Nope. North Korean threat actors continue to use malicious VS Code tasks files as many developers have re-enabled the feature, or other developer tooling has enabled it. In fact, most of the large-scale attacks we've seen so far in 2026 leverage VS Code tasks and settings files to help re-distribute the attack artifacts.

"The second observation is that the bad guys will find a way. Just because NPM won't run scripts at install time doesn't stop users from running those scripts the second those packages are installed. Even worse, if you already use a library, and it's compromised, you don't need to run install scripts to receive the payload. It's going to run in your application. No scripts needed. Now follow the incentives one step further. If install scripts are switched off by default, some package authors with legitimate needs will stop relying on them. Good ones will document a manual build step. Others will move the work somewhere NPM cannot turn it off - a cURL piped through bash in the install.js, a separate bootstrap binary, a setup command you run after installing.

"The attackers will make the exact same move, because of course they will. If the postinstall hook no longer fires automatically, you don't give up. You find the install path that still executes. This is precisely the kind of threat actor behavior I talked about at DEF CON. Push on one control, and the malicious activity relocates to where the controls aren't. It doesn't just vanish.

"And here's the part that should worry NPM but won't, because it's good for them. When the malware moves out of the registry and into a shell script on someone's gist, or a binary downloaded from a CDN, NPM gets to report that registry-resident malware is down." Yay. "They'll claim victory. The problem will look solved from the viewpoint of their dashboard. Meanwhile, the risk has simply relocated to terrain with less visibility and fewer tools, not more. The problem gets pushed off the one surface the entire security industry actually instruments, and onto surfaces nobody is watching. That's not a win. That's a measurement artifact.

"The third observation is that it gets more difficult for defenders then, not easier. Between cooldown periods and the disabling of install scripts, large-scale NPM attacks will become less frequent. But when you push legitimate functionality off the well-lit path, you don't just move it, you make it look guilty. A package author who genuinely needs to fetch a platform binary at install time, now doing it through some indirect mechanism to survive the new defaults, produces a fingerprint that looks exactly like evasion. Obfuscated loader, out-of-band fetch, install-time network call to a non-registry host. Five years ago that was a strong malware signal. After v12, a chunk of it is just legitimate software adapting to a stricter world." Not malware because of the things it has to do.

"So while the number and frequency of the big NPM attacks go down, the signal-to-noise ratio for everyone hunting malicious packages gets worse. The benign" - in other words, false positives, or missed negatives. "The benign and the malicious converge on the same suspicious-looking pattern. We end up triaging a flood of weird-but-fine packages to find the weird-and-actually-bad ones, and the bad ones get better cover precisely because so much legitimate behavior now looks like what they do. You bury the needed functionality in something that looks sketchy, and you've built the perfect place to hide a needle, a pattern that looks suss except it isn't. Until it is.

"What the firewalls and cooldowns are really telling you. Step back and look at what cooldowns and firewalls actually represent. They're good controls, yes. They're also an admission. The reason a third-party product can flag a malicious version six minutes after it's published is that NPM is not doing it themselves. The reason teams pay for an interception layer in front of the registry, the firewall, is that they cannot trust the registry to keep malware out.

"We are watching detection and response getting pushed onto users and onto a handful of vendors, while the registry that profits from being the default keeps under-investing in its own scanning. The incident cadence so far in 2026 makes this gap obvious. Version 12 is NPM catching up to the problem I laid out a year ago. Okay, that's progress. But a registry that has been this chronically under-resourced on internal security doesn't get to flip three defaults and call the supply-chain problem handled.

"So what actually does move the needle? Where does this leave us? We're left with the unglamorous truth that tooling is necessary and nowhere near sufficient. Turn on cooldowns today. It's the single best ratio of effort to protection available, and there's no reason to wait for v12. Prepare your approve-scripts allowlist now, on 11.16 and later, so the v12 upgrade does not break your builds and stampede your team into rubber-stamping everything. If you can manage package firewalls, run them. Do all of it." And that's his conclusion. So I thought this was a terrific...

Leo: There's my minimum release age, 14, for NPM and also do it on bun.

Steve: 14 days, good.

Leo: Bun you have to do seconds, I think. And I did it for a lot of things. I wish I could do it for the Arch user repository. I can't. I wish I could. Or at least I...

Steve: Yeah, that's be nice.

Leo: Yeah, because that's the one that's really scaring me right now.

Steve: Yeah. So I thought this was really a terrific piece of "from the field" feedback. As we've been seeing for years, malware that slips into the NPM repository has been steadily increasing, I mean, like big-time. And there doesn't appear to be anything that can be done. The one thought I had while reading this person's posting was that perhaps these changes are being made on the cusp of AI appearing at a time of need. He noted his annoyance that responsibility was effectively being pushed away from the repository and onto less centrally managed external resources and solutions. We see that the largest entities, Cisco and Microsoft, jump to mind. You know, they've been unable to even police their own internal closed source offerings to rid them of bugs.

So the scope of the task for an open source free-for-all repository should not be underestimated. I mean, I'm sympathetic. How do you allow anybody who wants to to contribute a package and have any security? But just as AI is now promising to revolutionize the quality of closed source offerings, it also seems like the perfect solution for repositories such as NPM. I think it makes nothing but sense. So we know that IBM and Red Hat are going to be pouring a ton of money, was it $5 billion, into using AI to help open source. And certainly NPM ought to be a big initial target for them.

Leo: Yeah.

Steve: And you know, Leo, it's time for me to take a sip of my juice and for us to find out who's paying for this.

Leo: This is a low-caffeine day for Mr. Gibson. What is in your juice? Is it a green juice? Is it a...

Steve: No, it's just it's a third of regular orange juice - organic, of course, because Lorrie - and then two-thirds water.

Leo: That sounds good.

Steve: So, you know, yeah, it's just diluted...

Leo: Diluted in the kidney.

Steve: It would be water except water, eh, it's a little boring. So we just, like, you know, just make it a little more interesting.

Leo: It's like Tang.

Steve: Exactly.

Leo: Okay.

Steve: Okay, feedback time. Roger Voegtlen. Best I can do. Sorry, Robert.

Leo: The "G" is silent. I think it's Voegtlen.

Steve: Robert Voegtlen. Hey, Robert. So get this, Leo. He says: "Dear Mr. Gibson: I started listening to Security Now! when I was in the fourth grade."

Leo: Oh, geez. Okay, Robert.

Steve: "I'm now 22 years old and have a degree in Cybersecurity from Augusta University."

Leo: Right on.

Steve: "Security Now! has always been there for as long as I can remember. As I've begun my job search, I've realized just how vast cybersecurity really is. There are so many specialties, the more I learn, the more I realize I know nothing. What I've been struggling with is figuring out where to focus. It feels like it could take decades of working across different roles and industries before I discover the niche that truly fits me. By then, much of my career will already be behind me. How do you determine which area of cybersecurity is worth dedicating your life's work to? Thank you and Leo for everything you've done through Security Now! over the years. Sincerely, Robert Voegtlen, SpinRite owner."

Leo: Aww. What a nice...

Steve: So first of all, Robert, wow. My question to you would be what could have possibly motivated you as a fourth grader to tune into this podcast? That's really something.

Leo: [Crosstalk] talk about zero days, Mr. Gibson.

Steve: You know, and it is the case that Leo and I have been here throughout your entire life.

Leo: Wow.

Steve: So anyway, I know that we both feel very glad and fortunate that we've been able to be here the whole time.

Leo: Absolutely, yeah.

Steve: To your question? I have no idea. In my case I sort of stumbled into the security sphere through ShieldsUP!, which I created because there was clearly a need for it at the time. And then the OptOut! adware remover, which happened when I discovered that Aureate adware was on my own PC. Since my background since about age five, began with understanding electricity, and then expanded into electronics and physics and engineering and computer hardware and software, I had the broad background ahead of time to pretty much head in any direction. And at the time when SpinRite was purring along, I saw a need over in the security space.

So I think my best advice would be to perhaps more deliberately do what I had inadvertently done, which is to obtain the widest possible background exposure that you can. That will automatically expose you to many more aspects of the field, and you may find that you naturally gravitate toward something specific. And even if that doesn't happen, I believe you'll be better equipped to succeed with whatever you decide to tackle. In today's highly competitive world, I normally advise people to become the best they possibly can at a narrowly targeted specialty. I believe that's where to succeed today. But of course that process of specialization can only begin once you've determined what truly interests you. Leo, any thoughts?

Leo: Well, I think it's - my kids ask me this, too. This is kind of the eternal question, one of the most fundamental questions every human asks.

Steve: And today there is so much happening, I don't know what direction I would take. I can really relate to Robert's position.

Leo: And even if it's not cybersecurity, if it's just whatever, you know, history or whatever, how to figure out what it is that you want to devote your life to is a very, very challenging question. But I think your answer is exactly right. All you can do is try as many things as possible.

Steve: Expose yourself.

Leo: And if you've found one...

Steve: See what sticks, yeah.

Leo: And just listen to your heart, and you'll know when it sticks. You'll know; right? Steve had no choice. He knew exactly what he wanted to do. He just did it, you know, and just like it clicks, and you're going to do it. I didn't, either. You know, we just - we did what we loved. And I think that's the best thing anybody can do.

Steve: Oh, to be able to spend your life doing what you love, I mean, I love that the best, shortest summary of that is to say "I never worked a day in my life."

Leo: Right.

Steve: Because it's not. It's joy.

Leo: Right. And I hope you can find that.

Steve: Yeah, exactly. Todd Whittaker, whose last name I can pronounce, is a college professor listener of ours who uses PHP to teach security.

Leo: Oh, boy.

Steve: Yeah. He wrote: "Hi, Steve. Your recent discussion of insecure PHP in Episode 1082" - last week - "rang very true to me. But you know what? PHP is really quite useful in cybersecurity, just maybe not the way its defenders usually mean.

"I'm a college professor, and when I built our undergraduate cybersecurity major back in 2010, two of the original courses used PHP: Web Programming and Application Security. Clearly, that was not because PHP represented our ideal of secure software design. It was because PHP is almost perfectly suited to showing students how insecure software

gets built. SQL injection? Concatenate user input into raw SQL. Cross-site scripting? Echo unescaped input back to the browser. Broken authentication? Store passwords badly or trust the wrong session variable."

He writes: "PHP makes the mistake easy to write, easy to see, and therefore easy to teach." He said: "Once students can see the failure clearly, we can show the corresponding discipline: prepared statements, output encoding, password hashing, access control, session hygiene, and least privilege. That, for me, is PHP's real value in a security classroom. It gives students a compact, legible catalog of the mistakes they need to recognize before they encounter them in the wild. Many cybersecurity graduates will eventually audit, inherit, or respond to PHP-heavy environments, including Drupal and WordPress installations, where they will need to see past code that merely works and recognize the familiar patterns that make it vulnerable. And if they leave the courses better equipped to be skeptical of PHP-based platforms in environments where security matters, so much the better. Todd."

Leo: Awesome.

Steve: So, yes, I think Professor Todd's use of PHP for teaching about coding security is brilliant. It's an application for PHP that had never occurred to me. I love it. So thank you, Todd.

Derak Kilgo said: "Hi, Steve. 20-year PHP veteran here. I thought you'd find this interesting: The foundation that governs PHP has added a grant-funded position to improve PHP's security posture given the realities of AI-powered development. "The PHP Foundation grant will fund a six-month full-time position titled 'Ecosystem AI Security Engineer in Residence at the PHP Foundation' to lead this effort and to prepare a sustainability plan for the time after this initial phase. This person will act as a trusted intermediary between security researchers and maintainers in urgent, high-risk situations, and will collaborate with peers in similar roles across other language ecosystems. Additionally, grant funding will also be employed toward the team goals described above where they cannot be accomplished by the single paid lead position or with the help of PHP community volunteers."

And he included a link to the announcement of the ecosystem security team. He finishes: "I've been listening since you were an actual podcast on an iPod with a spinning disk. Love the show. Your perspective is appreciated. Derak."

So I think overall the PHP project's move is great. But I should be clear, and Todd Whittaker's use of PHP for security education helps to further clarify this: I do not believe that there's anything whatsoever wrong with PHP. It's not at all the language itself I do not trust. There's nothing wrong with the PHP language. I think it's very likely bulletproof. My problem with it, which has been informed through our two decades of covering its use, is that those who often gravitate to PHP do so because it appears to be so easy to use. Because it is. But we've learned that it's always easier to write code that works than code that works securely.

This next bit of listener feedback sets up a perfect example. Steve Meyers wrote, and his subject was "Your PHP rant." He said: "It seemed like your rant against PHP was a bit of a stretch. PHP has its issues, but it's also come a long way. Here are my issues with your rant: You were using some obscure WordPress plugin with a whopping" - and he's not being serious here, sarcastic - "with a whopping 4,000 installs as the basis for complaining about PHP. Your specific complaint about PHP making it too easy to do bad things is that it has an eval() function. Here is a list of some other programming

languages with eval() functions: JavaScript, Python, Ruby, Perl, Lua, Lisp (specifically noted in Wikipedia is Lisp as the originator of the eval() function)."

Leo: Yeah, it's [crosstalk].

Steve: Yes. Yes. "Scheme, Clojure, and MATLAB. Anyway," he finishes, "it really seemed like your rant was pretty gratuitous and did not have a lot to back it up."

Okay. So first of all, Steve is of course correct that the particular problem with that PHP Wordpress add-on was due to its programmer perhaps not being aware of the danger of forwarding user-provided text into an eval() function. And it's certainly true that similar eval() functions exist in many languages. But I'm not upset with PHP at all for having an eval() function. My concern is that writing secure code for the web is extremely challenging. That's why through the years we've been covering all the mistakes that can be made, many not with PHP. There are so many different and very subtle ways to screw that up. Professor Whittaker's note mentioned a handful of them, and he didn't even mention eval().

So perhaps the best "non-ranty" way of expressing what I mean here is to say that it feels as if there's a larger inherent mismatch between the coding skill of the typical PHP coder, who may be coding for the web for the first time, and the coding skill required to do so securely, probably in a different language. It's certainly the case that no sane person will have decided to code their website in Lisp. Okay. But that said, I'm probably not one to speak since I did choose to code mine in assembly language.

Leo: Yeah. Which is worse than Lisp, so, okay.

Steve: Which is, yeah. Okay, Leo. Our last break, then we're going to look at Patch Tuesday la AI.

Leo: Ah, Lisp. I seem to remember programming in that language once, back in the day, before I started coding in English. Which also has an eval function.

Steve: Incredible. And the code that AI's producing for you is C, largely?

Leo: No. I let it choose its language. It's pretty evenly split. Go is my current favorite because it's concurrent, but Rust is great because it's so safe.

Steve: Safe, yes.

Leo: And so it's a little provable, you know, you can - a little easier to prove that Rust code is safe. And then Typescript is often chosen for similar reasons.

Steve: Right.

Leo: So almost all my code is either Rust, Go, or Typescript.

Steve: And the web has tons of examples for the AI to have trained on.

Leo: Exactly. Which is why I don't use Common Lisp, because it's a little less [crosstalk].

Steve: Not so common.

Leo: Not so common. Although, oddly, my AI is very good at my Emacs configuration, which is done in Elisp. It's really good at writing Elisp code. So, you know, there's a lot of that out there. I guess maybe more even than Common Lisp.

Steve: Yeah.

Leo: But Go is probably I would say a good choice. Python, oh, I left out Python. Lot of stuff's in Python. Just all those one-off - like that one-off web server for the curling of the SSH key.

Steve: That it just threw together for you.

Leo: So easy to do in Python because all the libraries makes it very simple.

Steve: Right, right.

Leo: Now, back to Steve.

Steve: So we all knew this had to be coming, and it did not take long, which is probably the mantra for the AI era. It didn't take long.

Leo: No.

Steve: Because the new race is now on, right, to see whether our industry's badly broken software can and will be repaired, with the help of AI, before the bad guys are able to leverage that same AI to find and exploit any of that same software that's not yet been repaired. In other words, what's been expected and predicted with the advancing evolution of AI models focused upon code is all really happening.

Last Tuesday, Microsoft broke their all-time record for the number of security vulnerabilities patched in a single update cycle. And that doesn't even count their fixes to their Chromium-based Edge browser, which also broke its own record by a lot. Those fixes to Edge are now wisely being separated into a separate count. And I say "wisely" because, if the two counts were not separated - and we used to lump IE in with, you know, Patch Tuesday updates because that's when it was being updated typically. If they weren't separated, June's haul alone would land somewhere in the neighborhood of 566 security vulnerabilities. 566.

Okay. So let's first - yikes. Let's first take the non-Edge Windows and Windows-adjacent patches. Even without Edge, there are so many that the exact number differs from one report to the next. Counting that high could be tricky. But everyone agrees that it was at least 200 and perhaps as many as 206. Now, we've all become a little patch-drunk; right? There was a time when 30 or 40 vulnerabilities would have raised eyebrows. Whoa, 40 vulnerabilities. Now that's seen as a quiet month. You know, it's like, 200. So consider, you know, stand back and consider something north of 200 security bugs found and fixed. On the one hand, it's great that Windows and its surrounding software will have at least 200 fewer discoverable security vulnerabilities. But on the other hand, Windows software had 200 or more security vulnerabilities to be fixed; and no one imagines that next month will be much better. I don't think they got them all. So, you know, buckle up for July.

And it's not as if these were insignificant problems, you know, that could have just been ignored. Oh, no. Among those 200 were six zero-days, five of which had previously been publicly disclosed, most by you know who, and one which was under active exploitation in attacks. And among these 200 now-blessedly-resolved Microsoft Windows and Windows-adjacent bugs, get this. 33 of those ranked as "Critical," with all but five of those, so 28 of them, being remote code execution. 28 critical remote code execution flaws. Of the remaining five, four of those were privilege elevation, and the last one was an information disclosure. But 28 RCEs found and fixed in one month.

It's nice to see Microsoft moving quickly to employ their Codename MDASH AI - I hope that doesn't stick - to the task of finding and fixing the many flaws we have come to understand Microsoft's code contains; right? I mean, they're moving quickly with a purpose here because they are fully aware that increasingly-capable AI models are also in the hands of malicious actors who are beginning to actively employ those models to the discovery and exploitation of Microsoft's many code shortcomings. So as I noted at the top of this, it could not be more true that a true race is on. This is really a race.

I continue to hold, however, that bugs are not inherently endless. Once removed, bugs almost always stay gone. There are some regressions from time to time. But mostly they stay gone. And if similar AI is employed eventually, hopefully already now, to pre-screen new code before it's released, the past's continuing supply of freshly created new bugs should finally also be cut off. This means that the consequences of this newly AI-enabled race, which is motivating both sides more than anything ever has, is that we're all going to be receiving far better software from Microsoft than we ever have. Basically it's sort of like - I remember noting this about Cisco, thinking, okay, Cisco. You've been unable to, like, get your software right. Maybe we need to make it easier for you. AI will make it easier for you. Similarly with Microsoft. AI is going to make it easier for Microsoft to have way fewer bugs than ever before.

Okay. So what's the overall breakdown of these 200-some vulnerabilities? Last Tuesday's more-worthwhile-than-ever round of updates fixed 65 elevation of privilege vulnerabilities, which we know those are just as important. They don't sound as scary as remote code execution, where the bad guys provide the code they'd like to run in your computer; but many times bad guys get in on a user account. So they need to elevate their privilege to get to really sink their teeth into the system. So 65 elevation of privilege vulnerabilities, 55 remote code execution vulnerabilities - 55! - 30 information disclosure vulnerabilities, 27 spoofing vulnerabilities, 19 security feature bypass vulnerabilities, and seven denial of service vulnerabilities, you know, where you crash something. One of those we know, one of those DOSes we will talk about in a second because that was the HTTP/2 bomb that we covered.

And just for the record, even all of that, 206, 200 or 206, depending on who you ask, they did not include flaws repaired previously in Mariner, Azure HorizonDB, Microsoft Copilot, Copilot Chat, M365 Copilot, Microsoft Exchange Online, and Microsoft Graph.

They were all previously repaired, and they all had a bunch. In other words, things really are quite furious up there in Redmond, Washington at the moment. And that's great for everyone.

So what do we know about the various zero-days that were fixed? There were six of them. Thanks to the tireless efforts of the renegade hacker Nightmare Eclipse, another of their zero-days known as GreenPlasma was fixed. That was assigned the CVE this year of 45586, an elevation of privilege flaw that the disgruntled hacker discovered in Windows Collaborative Translation Framework (CTFMON). I don't know what that is, but I've often seen CTFMON in my process list so it's busy being something. It had been publicly disclosed and enabled an unprivileged user account to upgrade itself to full system privileges. So again, Nightmare Eclipse was another nightmare for Microsoft, publicly releasing as a zero day a means of using the CTFMON to elevate an account privilege from user to system.

Microsoft was also able on June, and I was impressed by the speed of this because this only just happened earlier in June, to quickly repair that HTTP/2 Bomb, which we just talked about, denial-of-service vulnerability, which was, you know, the deliberate headline-grabbing irresponsible disclosure of which annoyed me so much, which we talked about last week, where the guys at Calif said, hey, guess what, we don't believe in responsible disclosure anymore because of AI. So here it is. Okay.

The Microsoft variant was assigned a CVE of 49160 and was found in the HTTP.sys module, which makes sense. That's the web server module, or driver. As Microsoft phrased it: "Uncontrolled resource consumption in HTTP/2 allows an unauthorized attacker to deny service over a network," which of course is Microspeak "for anyone's laptop can bring down our web servers." What we already know from Calif's disclosure is that the HTTP/2 Bomb is a denial-of-service technique that abuses how the protocol itself compresses and manages web traffic headers, which allow attackers to send very small amounts of data and force servers to allocate disproportionately large amounts of their memory. Then by combining two techniques, the basically not bugs but just protocol features, the researchers at Calif discovered that they could dramatically increase server process memory consumption, then keep the memory tied up by manipulating HTTP's flow-control settings to prevent the server from freeing the resources. Basically don't allow the inbound query to ever end. So the server just waits, and it waits with 32GB of memory tied up. And the laptop keeps doing those until all the server's memory is gone, and it crashes.

So since this clever attack is more an abuse of deliberate HTTP/2 protocol features rather than a bug that can be fixed, Microsoft also added a new "MaxHeadersCount" registry setting to limit the number of headers in a single request. If it's not specified, the default maximum header count is 200. It can be set as low as 50 or as high as 65,535. So, you know, of all 16 bits turned on in the count.

Tuesday's updates also resolved two problems with BitLocker: Nightmare Eclipse's so-called "YellowKey" vulnerability - remember we talked about this, that was the wacky boot thing - that's been addressed as CVE 45585. That was the hack that involved rebooting a machine while supplying a script on a thumb drive that had the effect of deleting some files and leaving the system in a pre-booted state with its primary drive decryption key still loaded, and the normally encrypted drive, the main system drive, fully decrypted and accessible.

As Microsoft phrased it: "A successful attacker could bypass the BitLocker Device Encryption feature on the system storage device. An attacker with physical access to the target could exploit this vulnerability to gain access to encrypted data." So after last Tuesday's updates, this can no longer be accomplished. Microsoft is now careful to not leave the BitLocker-encrypted drive in an unencrypted state.

So what about the second BitLocker repair? Let's hope that the way they stumbled on this one was human-derived and not AI, since it sure feels like "the old Microsoft," as you'll see in a minute, rather than the new and improved Microsoft we're hoping AI might be enabling. The second flaw was named "Bitskrieg," cleverly.

Leo: Oh, Bitskrieg, yeah.

Steve: Yes, by the guy who discovered it. He originally posted the news of his discovery over on X under the name "JonasL." Since the view from a hacker's perspective is always interesting and often entertaining, I'm going to share Jonas's original posting which was near the start of the month, on June 4th. And again, as I said, I was impressed that Microsoft moved this quickly. Hadn't occurred to me before, but I wonder if AI might be accelerating the pace from them having knowledge of the problem and it being fixed. Or maybe the confidence. Because, after all, that was five days from June 4th to June 9th. So that's very quick.

So JonasL wrote: "It is rare that anything new happens in the world of IT security. It's mostly just an endless cycle of variants of the same vulnerabilities being exploited over and over again. That's why I appreciate when something new happens; and the YellowKey exploit was, for once, an attack I had never seen before."

He said: "I've myself done a BitLocker bypass before." Then he supplies us with a link which has an embedded January 15th, 2021 date, so about five years, a little more than five years ago. He said: "But this one was new to me." Meaning YellowKey. "It expanded the attack surface onto an area I had not looked into before, the recovery environment. The recovery environment is stored on a partition that BitLocker does not encrypt, and the TPM is not locked until you use any functionality that's not the startup repair. So if you somehow get code running without causing the TPM to lock, you have access to the encrypted drive.

"Microsoft killed YellowKey by removing the auto-execution of a newly introduced component that could be manipulated into doing file operations by rolling back a transaction stored on a USB drive." Which was really clever, by the way. "I suspect they simply copied the recovery environment from the unreleased Windows Cloud made for thin clients. That also explained why the bare metal recovery EFI image identifies as Windows 365 when downloading what to boot on its RAM drive from the Microsoft server."

He said: "The YellowKey transaction rollback hack enabled a file deletion, enabling launching a command prompt by holding down CTRL when launching recenv.exe," which he also agrees is an elegant attack. "So when my friend asked if I wanted to try to help restore the vulnerability, I figured why not give it a try?" He says: "Microsoft fixed YellowKey by just killing a specific vulnerability, but they did not resolve the underlying design issue, so I'd be surprised," he wrote, "if it wasn't doable. After 24 hours, a new attack was born. I call it 'bitskrieg.'"

Okay. So his posting then walks us through his successful hack and attack which demonstrated that there was, indeed, more than one way to skin this particular cat. And I'll note again, as I did before, that vulnerabilities in BitLocker access at this point is an inherent weakness that Microsoft really cannot do much about. Sure, they could be much less sloppy and more carefully consider the consequences of their design decisions. But if we want a system that has its BitLockered main drive encrypted at rest, which then autonomously boots into the Windows OS environment that's contained within the BitLockered drive, without requiring some information that is NOT stored on the local machine, you know, which is where the user-provided PIN comes in, then there's really

no way around the fact that the machine will be vulnerable to some form of local-access boot-time shenanigans. There's just no way around that.

So last Tuesday's patch update resolved this additional "bitskrieg" attack that JonasL discovered. And as I said, I'm surprised they did it in five days. And that's good, since he had made it public also. But enterprises and security-minded end users should not rely too heavily upon the security of entirely self-decrypting BitLockered systems. The only way to ever be truly safe is to require some information at boot time that is NOT present anywhere else in the system. That means depending upon a hardware dongle or a manually entered PIN. This is the classic tradeoff between security and convenience, and there's just no way around it.

Okay, so what else do we know about Tuesday's record breaker? The history behind this next zero-day is curious because it's a fix for a CVE dated an unbelievable six years ago in 2020. CVE-2020-17103, which is a Windows Cloud Files Mini Filter Driver Elevation of Privilege Vulnerability. Like so many other recently released zero-days, this one, too, owes our attention to none other than Nightmare Eclipse. Its reincarnation by that now-infamous hacker was given the name "Mini Plasma." So, yeah, Mini Plasma.

So what's up with the CVE from six years ago? Nightmare Eclipse explained that the flaw was originally reported to Microsoft by Google's Project Zero. And, indeed, I found the original bug with its full description and its attached proof of concept in a ZIP file. However, Nightmare Eclipse stated that the flaw was still exploitable, and it was unclear whether Microsoft fully patched the issue or whether the bug may have been reintroduced, as I said, bugs sometimes come back. May have been reintroduced at some point. In any event, it appears to have been fixed once again.

And this brings us to the fifth and final zero-day remote code execution vulnerability. Just to be clear, this is one of the 55 remote code execution vulnerabilities that were fixed last Tuesday, though the majority, while RCEs, were not zero-days. Remember there were 55 of those, unbelievably. But this one, CVE-2026-42897, was one of the zero days. This last one is a spoofing vulnerability that was present in Microsoft Exchange Server. It was being actively exploited in the wild to execute JavaScript in a targeted victim's browser. Microsoft explained: "An attacker could exploit this issue by sending a specially crafted email to a user. If the user opens the email in Outlook Web Access and certain interaction conditions are met, arbitrary JavaScript can be executed in the user's browser context."

So technically, this last one was not part of the primary patch Tuesday bundle. At the time, Microsoft explained that they were still working on its update and that they would be pushing it out through the Exchange Emergency Mitigation Service, which should be enabled by default, certainly on Exchange Server, and hopefully for users that might be affected by this. We know nothing about who disclosed this vulnerability, nor how it was exploited. But it was a zero-day.

Scanning down the surprisingly lengthy list of CRITICAL - so remember we had 55 critical remote code vulnerabilities - we find that one was present in Active Directory Domain Services and another in Microsoft Azure Kubernetes Service. Microsoft Office and the Remote Desktop Client - get this - each had seven remote, critical remote code execution vulnerabilities. Remote Desktop and Microsoft Office each had seven. One was in Nuance PowerScribe. Three were found and removed from Windows Hyper-V, thank goodness, because I'm about to start using it. There was one in Windows Deployment Services. Windows DHCP Client had one. And that's a scary place to have a critical RCE, although it explains why it's critical, since most Windows clients will be using DHCP, so their client will be vulnerable remotely by default to any server that is able to supply DHCP information.

In addition to the Windows HTTP/2 Bomb, which we talked about last week and previously this week, which was that denial of service problem, the HTTP.sys module also had a terrifying critical remote code execution vulnerability fixed. I assume it was terrifying, since it was in HTTP.sys. That's the Windows web server module, and it was rated critical. And it was a remote code execution. And nothing is more exposed than a web server, which is why the HTTP/2 Bomb vulnerability was such a problem. Windows Kerberos also had a critical RCE; and, not to be left out, Windows Kernel did, too. And finally, two RCEs were found and fixed in the Windows Win32K Graphic GRFX module.

So, whew. Yeah. I named today's podcast "Patch Tuesday la AI" because this is what the next several months or so are probably going to look like. It's going to be very interesting to see the shape of the vulnerability discovery and remediation curve. As we know, the previous month's, meaning May Patch Tuesday, tied for the most ever patches in any month. And that was like 125 or something. As we know, this month's patch number, count, fixes, far exceeded that one. So are we seeing an acceleration? I mean, we are in the last couple of months, certainly. What will next month's look like? No idea. But it's going to be really interesting to see.

And finally, lastly, over on the Microsoft Edge/Chromium updates side, I will quote from BleepingComputer's reporting about that, just one line, which wrote: "There were also a massive" - their word. "There were also a massive 360 Microsoft Edge/Chromium flaws that were fixed by Google this month." Okay. 360! And these were found in the Chromium browser which was already the recipient of an incredible expenditure of past manpower. But now our entire software industry is replacing manpower with AI-power as rapidly as it can, and it's quickly becoming clear that at least in the field of software, there's really no comparison, man versus machine. Since the results are pretty much speaking for themselves, machine wins.

Leo: Yeah.

Steve: And I, for one, cannot wait to see what happens next. This is a real ride lately.

Leo: It's crazy that there are that many flaws in Chrome and Windows. Windows is huge, I understand. But Chrome.

Steve: Yes. And it does say that Google is busy running AI over Chrome.

Leo: Yeah.

Steve: No, but I'm with you. I am stunned because, I mean, it was believed to be super secure. And they said, uh, whoops.

Leo: So in some ways this, I mean, if you could find the flaws, if you were a bad guy, you could also exploit the flaws; right? I mean, that's why people are worried about Fable and Mythos is finding them and fixing them is tantamount to finding them and exploiting them. It's the same process.

Steve: And Mythos we know produces proofs of concept.

Leo: Right.

Steve: So it designs the exploit. That is an exploit. It's a proof of concept.

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/