A Strategic Plug Pull

Description: We take a detailed look at the many facets and aspects of last week's wild PHP ride: What happened. How it happened. Why it happened. How it could have been prevented. How it impacted many of our listeners. We consider the implications of what is said to be coming next and the idea of temporary "strategic disconnection" from the Internet.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-1102.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-1102-lq.mp3

SHOW TEASE: You've been waiting for it, and now it's here. Coming up on Security Now!, Steve Gibson gives you the lowdown and the uptake on PHP. Yes. Should you be worried? Who is involved? And how did they find out what was going on. There's so much to dig into. Plus we also get some great listener feedback this week, a hilarious conversation about AI and poison mushrooms, if you can believe it, and the strategic disconnect as Steve advocates for a safer web. All of that coming up on Security Now!.

MIKAH SARGENT: This is Security Now! with Steve Gibson, Episode 1102, recorded Tuesday, October 27th, 2026: "A Strategic Plug Pull." If you're here and you're shivering, I understand. I mean, shivering in fear. It's time for Security Now!. And yes, it is finally time to hear what's going on in the world of security this week as we grapple with far too many issues, concerns, bugs, fixes, patches, et cetera, on the show. Of course I am Mikah Sargent, subbing in this week for Leo Laporte. And we have the wonderful Steve Gibson. Hello, Steve.

Steve Gibson: Mikah, great to be with you again.

MIKAH: So good to have you.

Steve: For our second of three weeks together; right?

MIKAH: Yeah, yes. It's sad, the countdown is almost over. But I've been looking forward to this week for sure because I know we've got a lot to talk about.

Steve: Well, yes. It was the day after we did last week's podcast, which was Tuesday the 20th. On the 21st, the you-know-what hit the fan. Yeah. It's been a crazy time. So we're going to take a look at this week, pretty much this podcast dominated by - there are some other things, but dominated by this first massive event in what is clearly AI-enabled. There are so many things to talk about. But, for example, this bug was known for many years.

MIKAH: That's what really upsets me.

Steve: Yes. And it was like, well, yeah, you know, it's PHP. What are you going to do? But it wasn't known to be exploitable. That's the thing that AI brought to bear. And then the other cool thing, or really, well, I mean, okay.

MIKAH: Cool.

Steve: Take a breath, Gibson, here. Okay. We're going to take a detailed look at the many facets and aspects of what happened last week with the industry's wild PHP ride - what happened; how it

happened; why it happened; how it could have been prevented (because it was preventable); and, interestingly, how it impacted many of our listeners. I sent out a special mailing Wednesday, late morning, to all the subscribers of the Security Now! podcast. I called it a "Flash Alert." Just to make sure...

MIKAH: I thought the sky was falling.

Steve: Just to make sure that everybody got the message. I mean, certainly enterprises are going to be on top of the news, we hope. But still, that generated a bunch of interesting replies from our listeners that I'm going to share. Also we need to consider the implications of what is said to be coming next. Now, and what's interesting, as we'll see, this is from - there are people in the know, people who just as last time, that is to say, last week, had been keeping a secret for two months because this was discovered in August. And so August and most of October, I mean, I'm sorry, September and most of October, this secret had been kept. As we know, it leaked out a day or two early. But secrets are hard to keep.

The point is the people who were keeping the secret, some of them, are also keeping other secrets. That is, this is not a "will this happen again" or "what if." It's no, it is absolutely when we don't know or what we don't know. But we know that there are other things that have already been found, much like PHP's catastrophe, which haven't yet been disclosed. So anyway, as I said, lots to talk about. We have a fun Picture of the Week. We're going to look at listener feedback. And I called today's episode "A Strategic Plug Pull" because I want to propose that the idea of a temporary disconnect from the Internet may be the right strategy, depending upon what happens next. We don't know. What we do know is that it was the right strategy for a lot of Europe last week, who just said, whoops, let's go dark while we patch this, because there was that much concern.

Anyway, great podcast for everybody. And by the time we're done, everyone will have had their fill of PHP. We're going to look at, like, all of this. But what's interesting is that we don't know what's coming next, but we do know something is. There's more.

MIKAH: But wait. There's more. Great.

Steve: I'm going to handle our Picture of the Week a little differently this time. Normally I lead with the headline, the caption that I have given whatever the picture is. This time I'm going to follow with it. We have a two-frame cartoon for our Picture of the Week. The first, we see a camper, he's got a backpack on. We see a little trail sign behind him, a picnic bench in the background. So, you know, he's stopped, and he's asking his AI, "Is this mushroom safe to eat?" And the AI snaps his fingers and says "Yes! It's safe to eat!"

Second of the two frames of the cartoon we see that, well, this camper has not fared that well. His eyes are X'd out. His tongue is hanging out. He's dead. And in fact he's leaning against a tombstone with an RIP, you know, Rest in Peace, on it. He's not saying anything. However, the AI is responding, "You're right. That mushroom is poisonous. I'm sorry for the mistake! Would you like to learn more about poisonous mushrooms and how to identify safe ones?"

MIKAH: Sounds about right. Absolutely.

Steve: And that's just it, is it's an experience all of us who've had who have been using conversational AI. It's much less often happening today than it was before. Conversational AI has been getting better about checking its facts. However, the caption that I gave this was "It's not a bug. It's an inherent artifact of LLM technology." Which is what I believe. I'm going to at one point at the end of our listener feedback section, responding to a piece of listener feedback, I'm going to spend just a couple minutes because it's actually a conversation that I was having online in GRC's forums, sort of summing up the issue of consciousness versus intelligence, and responding to that question.

I know that many of our listeners have been saying to me, yes, don't stop with the AI. It's important. It's what's happening, blah blah blah. At the same time there have been another set of listeners who are like, enough with the AI already. So for what it's worth, I can see this beginning to wind down, that is, this podcast's coverage of all things AI, not because AI is less important than we thought, not because it's going away, none of that's happening, but because we've pretty much covered all of the various aspects and basis of it to my own satisfaction. You know, more will be happening. The big

events which appear to be coming, earth-shaking. This is all as a consequence of AI. But I'm going to assume that everybody who is following the podcast has listened to the last couple months of AI-centric focus and has taken from it what they're going to.

I do think that I'll be spending a lot less time on it because we've said a lot of what there is to say. If things change, of course, we'll do that. But, you know, today is all about PHP and the events, the security events of this past week. And I think we're going to be seeing more of that focus return. So just, you know, everybody, don't worry. We're not dropping AI.

MIKAH: It's tough, right, Steve, like the - because I have this, look, Leo's going to AI out the wazoo, and he's fine with that, and that's okay. That works for him, and he has shows to do that. I have a show called Tech News Weekly. And I see it as part of my responsibility to cover other things going on in tech. And it is regularly very difficult to find anything that is not also about AI. But I think we're both doing our best, and it sounds like you and I are both listening to all of the feedback for the people who are saying, no, I don't want that, and people saying yes, I do, and covering it from the perspectives of - through the lens of what we know.

And so that's something that I've appreciated about your way of covering it on this show is that it's never - you never take it outside of that lens. And I think that's what keeps it realistic and interesting and outside of the [exclamation], if that makes sense.

Steve: Well, and I think the other thing, too, is that without a doubt, the emergence of AI as it has impacted the cybersecurity space, it changed this podcast.

MIKAH: Right.

Steve: And whenever something changes, you're going to have some people who say, yay; and other people who say, hey, you changed the podcast. It's like, so that's inevitable; right? So I just wanted to say I feel as if I've gotten it out of my system. But I've said my piece. I've explained what I believe is what this all is. And it took a couple months, both to explain it and for me to internalize it and to think about it and read everything I could and to come to what I believe is an understanding of what we have. And so it's time to continue with, you know, security-related things and, exactly as you said, they will be AI adjacent because AI has hugely impacted the cybersecurity space.

Okay. So we're going to take a break from this podcast's longstanding tradition of eventually getting around to its main topic because there's nothing more I want to talk about than what happened last week with PHP. I mean, I'm not sticking it at the end is my point. There are so many different interesting aspects and sides to the story which surrounds last week's truly earth-shaking events. And the first good news is that it was earth-shaking, not earth-shattering. You know, the earth was not shattered. Even though this was a huge thing, we're all still here. But it was - the earth was certainly given a good shaking. Before we get into the after-the-event debrief, I just want to spend some time catching everyone up on the timeline surrounding what happened.

The bugs that enabled last week's pre-auth, meaning pre-authentication, meaning anybody anywhere on the public Internet was able to do this to any other PHP-based anything, you know, forum, NAS, web frontend, anything that had PHP in its default configuration since v5.4 when this began, which has been - that's way back there, all the way until last week's updates, thus pre-auth - was able to execute code on that targeted victim machine. That 5.4 release was March 1st, 2012, so 14 years ago. Every release of PHP for the past 14 years. As I said, until last week's multiple fixes to the various v8 branches, they're all vulnerable to this. They've always been vulnerable to this.

Now, we've talked about PHP often in the last 20-plus years of the podcast. It's suffered from perhaps more than its share of serious security vulnerabilities through the years. And at this point in my recounting of PHP's past troubles, this is where I would usually back up that statement of it's suffered from more than its usual share of troubles by enumerating them. Like, well, let's look at those, to put a point on it. But as I started to do that, the list became so long, literally, that I realized, okay, there's no point. No one's arguing that PHP has been an ongoing security trouble. I mean, nightmare, disaster, catastrophe, pick your adjective. Everybody listening to this podcast knows that to be true.

One tidbit I will share from the list of previous PHP nightmares was to recall how many problems, and I saw this again, PHP has suffered from its unserialize function. We've talked too often about the

problems with deserialization. The idea is that a language can be asked to save a complex object, which is typically represented in memory as structured data, as a data structure, to save that in memory data structure in order to store it. The object must be converted from its structured data into a "simple" - and I, you know, "simple" in quotes - a byte stream using an object-aware process which we call serialization. That's the easy half of the problem. It's reversing the process where it is so easy to fall down. And most implementations do fall down for a while.

Turning a previously serialized byte stream object back into the object it represents requires an interpreter to read the byte stream, figure out what it means, and then rebuild the original object. It turns out that doing this safely in the face of deliberate corruptive intent turns out to be surprisingly difficult, and we've covered many other instances in other languages. It's not just PHP that had a problem, although its problems were serious. It's had more trouble in this regard than probably any other language system. Its well-deserved reputation for security vulnerabilities, which I was covering for all of our listeners, with all of our listeners on this podcast for many years, it created a dilemma for me when I wanted to create web forums for GRC.

Up to that point, and that was only a few years back, I had handwritten every bit of application software running on GRC's server infrastructure myself, and of course all in assembly language. But all of the forum software I was considering sat on top of PHP. It was crazy for me to write my own. I would have loved to write my own. But if I had done that, I'd still be doing it. So bad idea. PHP is where all the forum software is. And as I mentioned on the podcast at the time, and I have many times since, there was no way I was willing to have a publicly exposed machine, which any forum has to be if it's going to be a public forum, running PHP within GRC's server infrastructure. Just too dangerous.

My solution was to spin up an entirely separate hardware server and sequester it from the rest of GRC's network behind a dedicated hardware firewall. That's what I did. And doing so at least would limit any damage that a PHP-facilitated compromise of that server might cause. Since then, I've added GRC's PHP-based NuevoMailer to do our mailing backend. That link shortener I'm often referring to, grc.sc. That's a little PHP app that I found somewhere, and I thought, yeah, it looks good. I don't really have to trust it that much because it's protected. For much of the web, even today in 2026, right now, and this is what we saw, right, we saw legacy, but we all saw current. PHP is a necessary and painful reality upon which a huge number of systems depend. Currently. I mean, like the latest version of - I mean, I use XenForo as our networking, as our web forum software system. All PHP.

So I want to just - I do want to take a moment to back up, just for the benefit of those who may not realize just how prevalent PHP has become and still is today. So right now, as we know, it's always been a popular language for web content management systems. So WordPress, Drupal, Joomla, TYPO3, Craft CMS, Concrete CMS, and Grav are all using it. PHP is the choice for several prominent eCommerce systems: WooCommerce, Magento/Adobe Commerce, PrestaShop, OpenCart, Shopware. The most popular Wikis: MediaWiki, which also runs Wikipedia, as well as other wikis, and DokuWiki, all authored in PHP. My forums, XenForo. Also obviously phpBB. What do you think that was written in?

MIKAH: It'd almost be faster to say the things it isn't.

Steve: Yeah. I mean, vBulletin, Invision Community, Flarum. In education, Moodle, which is the dominant open-source learning management system. Both Nextcloud and its predecessor ownCloud are PHP-based. We've talked about them in the past and recommended them. The classic MySQL/MariaDB admin interface named phpMyAdmin, what do you think that was written in? Both the famous cPanel and Plesk were implemented in PHP. It's the favorite CRM implementation language hosting SuiteCRM, SugarCRM, Mautic for marketing automation, Dolibarr for ERP and CRM.

Also large frameworks have PHP under them: Laravel, Symfony, CodeIgniter, CakePHP, Yii, and Laminas, which was originally named Zend Framework. All of that. And of course less universal systems. Apps. Lots of apps, like my NuevoMailer, which is the one I chose. You know, it's pretty much everywhere. So, you know, we heard the quotes, right, last week? At least 70, seven zero, 70% of the Internet has got PHP somewhere behind it, powering it.

So leveraging, unfortunately, the new power of AI for vulnerability discovery and exploit generation that emerged this year, the concern is that bad guys would be able to do this. The good news is

Google's Wiz Security and Calif Security independently discovered and privately reported to the PHP Foundation the critical vulnerabilities which were behind everything that caused such havoc last week. Both groups, both Google's Wiz Security and Calif, knew that too much of the Internet was using PHP to make light of a vulnerability that has been present and exploitable in PHP's default configuration for the past 14 years. So all 70%, essentially, of the Internet was vulnerable. And they kept that secret for two months while the project worked to solve this problem.

MIKAH: Can I ask, before we...

Steve: Yeah.

MIKAH: They kept that secret for two months. To me the scale of that and sort of just the idea of it feels rather difficult.

Steve: Monumental?

MIKAH: Yeah. Because you would have people in theory who would go, oh, but my people need to know, even though your people said, what is that like? Like how do security researchers decide and stick to that kind of thing?

Steve: Yeah, I mean, it is - that's a great question, and it really is their true commitment, and also, as they say, you want to keep a secret, don't tell anybody. So it is keeping it closely held, even within their organization. I'm sure there are people adjacent to the few who knew who are really pissed off, like...

MIKAH: Yeah. Like what, you didn't tell me?

Steve: Exactly. Exactly. How could you not tell me? Of course you could true me to keep your secret. Actually, we'll be talking about this a little bit later. There was a group who did know who were unable to keep their own secret as long as they promised to. I have a strong feeling they will not be in the inner sanctum next time.

MIKAH: If it's who I think it is, I read about that, and I thought, now, what were y'all doing?

Steve: Are you thinking Chainguard?

MIKAH: Oh, no, I guess it wasn't that. I thought I saw Cloudflare release stuff earlier than agreed-upon time.

Steve: That I hadn't seen. And I'm surprised because Matthew Prince is the person who, as we'll be seeing a little bit later, really does know. He's in the center of this.

MIKAH: Yeah, and of course this was all in the middle of it, so it's possible that that got updated and changed. But at the time I thought, wait, what? They moved too quick. But anyway, yeah. Continue, because I'm eating up this story.

Steve: So one of the things that you're reacting to, which is exactly right, it is incredibly difficult to deal with something on this scale. Not only how do you keep a secret, but we don't have in place the mechanisms for handling something like this. I mean, where the secret is going to be kept until a disclosure which has to be public and widespread, which means everybody learns about it. There's no way to only tell the good guys, is my point. And for something open source like PHP, there is no need to reverse engineer the binary to figure out what was done, which really does slow down the bad guys' ability to exploit it. All you have to do is do a diff on the source files from before and after. You immediately, as bad guys, see what was changed and then know how to go about working to make the exploit happen.

So in the case of open source it is really difficult. And we just don't have a technology in place. I'm stepping on a lot of points that I make later on because there is an exception to that which is, of course, our browsers. And there's a reason why they are an exception. Anyway, last Wednesday - actually, you know, we're a half-hour in. This is a great place to take a break. And then we're going

to look at OpenAI's sort of stepping into this because they had a role in it because it was GPT that was being used by the Calif guys.

Last Wednesday, OpenAI whose, as I said, their GPT AI was used to make this recent discovery using their "Daybreak" and "Patch the Planet" programs. They posted about all this whole event under their headline: "Patch the Planet: Global response coordination as agents uncover critical vulnerabilities." And, you know, their posting's naturally somewhat self-congratulatory, but it contains some important information which will be useful for navigating what could be the next several months or several years of AI-driven vulnerability discovery and patching. So I want to share that they said.

They wrote: "Over the past two months, we led a coalition of maintainers, researchers, and security providers to address critical vulnerabilities in PHP." And to sort of set the stage, they said: "An open-source programming language used to run websites and web applications. Calif identified a critical flaw using GPT-5.6 Sol as part of our Patch the Planet initiative, with Wiz also independently discovering and reporting both vulnerabilities. PHP's maintainers led the fixes and release, with our teams and partners like Wiz, Cloudflare, and others contributing research, technical support, and coordination to prepare protections ahead of disclosure.

"For organizations using PHP, the priority now is applying the fixes. We're sharing a PHP version checker as a Codex skill to help operators identify potentially vulnerable PHP versions and, alongside PHP's and their providers' guidance, determine which updates apply.

"Given the severity of these flaws and PHP's widespread use, we consider this one of the most impactful security issues in recent memory." And I would argue that's being modest. A lot of the other reporting said, you know, the most significant Internet security event ever. So OpenAI said: "It highlights how AI can uncover security flaws with Internet-scale impact" - as I said, a little self-congratulatory, although not their AI, any AI, and that's true - "but the same capabilities can also help attackers rapidly develop offensive tools and exploits. To our knowledge, frontier AI found these vulnerabilities before attackers did, opening a Defender's Window for maintainers and security providers to prepare fixes and protections ahead of disclosure.

"To help more defenders use these capabilities, we've committed $1 billion in subsidized Daybreak access through Daybreak for Frontline Defenders, focused on organizations with limited security resources protecting essential services.

"As significant as they are, these bugs are among several significant vulnerabilities we've already found and are working to fix, and we expect similar findings to increase sharply in coming months."

Because again, I want to highlight that. We're going to hear something similar a bit later directly from Cloudflare's Matthew Prince, the CEO. What becomes evident is that this PHP secret, which was closely held for two months before it leaked two days before its planned release date, is just one of the apparently critical discoveries that have been made and are currently embargoed as fixes are being prepared and coordinated, whatever that means. And we don't know what that means because we don't know what's being coordinated or fixed. But we're going to. So that makes what follows, I think, all the more important.

OpenAI continues, writing: "Drawing on our experience thus far, we're sharing a proposed approach, appropriate for the AI era, for responding to critical vulnerabilities in widely used software, and we're asking the security community to help improve this framework." So this sort of forms an open letter. "It brings together our models, security expertise, and maintainers to help turn findings into protection and build a response that can scale as AI capabilities advance. Getting a fix into use can mean adapting it for older releases, preparing provider mitigations, and updating systems across many organizations. A repeatable process helps these teams act quickly as more vulnerabilities are discovered.

"Through Patch the Planet" - which is, you know, the name they've given to their open source group effort in the same way that Anthropic has one. They said: "Through Patch the Planet we contributed security expertise and model access to support the PHP response, and worked with maintainers and research partners to test fixes and prepare guidance. We also helped coordinate with infrastructure providers on protections for affected systems and engaged government cyber agencies to help guidance reach critical organizations. This supported the maintainer-led response" - meaning the PHP Foundation response - "and helped other defenders prepare.

"For some findings, researchers can work directly with the software's maintainers to investigate the issue and develop a fix." And I would argue that's probably primarily what goes on. However, OpenAI's trying to keep their finger in the pie. So they said: "Others need a wider response because they affect many products or because distributors and service providers need time, which may be in short supply if frontier capabilities are not too far ahead of those that are broadly available, to prepare updates and protections. In those cases, the proposed framework can help maintainers identify who needs to be involved and coordinate the work. The response can involve five stages." And again, I'm still not really being sold on the idea that OpenAI has any role here other than the service provider of the AI, which is then leveraged to discover the problems and then fully understand them. But okay.

They said: "First, validate and fix the upstream vulnerability. The maintainer, researchers, and essential technical partners confirm the finding, assess its impact, and develop and test a fix. This process includes adversarial testing with frontier models to look for ways around the patch and related variants of the vulnerability." So again, yes, need to use AI for that. Makes sense.

"Second, give affected organizations time to act. We propose a framework for considering which organizations need advanced information to prepare protections, while maintainers decide whom to involve. We help with outreach, information-sharing agreements, and coordination between teams. A distributor might adapt a fix for an older release, a hosting provider might test a mitigation, and a security provider might prepare detection or prevention measures, including web application firewall rules." So I guess they're sort of saying there may be some overflow from what the software package maintainer is able to handle with whatever limited resources they may have. So OpenAI is saying we're big and commercial, and we'd be happy to lend a hand. Fine.

Third of five. They say: "Coordinate with government partners to help protect critical systems. Government agencies play an essential role in coordinating a wider response and helping security guidance reach organizations responsible for critical infrastructure and other essential services. In the case of PHP, we worked, in collaboration with our partners, to engage with multiple parts of the U.S. government, including the White House's Super Intelligence Force, the Office of the National Cyber Director, the Cybersecurity and Infrastructure Security Agency (CISA), as well as international partners to support that effort." So, okay, again, it makes sense if the PHP maintainers did not have themselves all of those direct contacts that an organization like OpenAI certainly is able to make those, you know, make that happen.

MIKAH: Yeah.

Steve: "Fourth, prepare updates across affected products. As the fix, guidance, and disclosure plans become stable, maintainers can involve additional distributors, hosting and service providers, response teams, and operators to prepare updates and address gaps in coverage."

And then, finally: "Five, once a patch is available, publish guidance and support deployment. Public guidance helps people assess their exposure, find authoritative updates or interim mitigations, and check that a fix is running." And as they said, they are providing a Claude Codex app for verifying PHP fixes. They said: "The response continues as operators deploy updates and report where they need." And I was thinking about your question about how this leaked or why, like a couple days before. How the secret was so well kept for two months and then, finally, shortly before.

What probably happened is that more people did need to be brought in in the final ramp-up in preparation for the full planned disclosure on Thursday, and that was the problem is that, as the secret, you know, sphere widened, then somebody was brought in who said, well, with all these people now knowing, no one will know if it was me who leaked.

MIKAH: That makes sense. Another thing here that I'm hearing is you're talking about all of the stuff that ChatGPT or that OpenAI is offering. It almost feels like the company just needs to or wants to appear as though it is being as helpful as possible in all of this. And so even if it's kind of like, you're not, as you said, it's not 100% clear that what they're saying and what they're offering is even helpful yet, up to, you know, the points of AI being involved, but yeah. If they've got the extra resources to throw around, they'd better look like they're doing what they can.

Steve: Well, and although it's no one's fault, it was their AI that uncovered this.

MIKAH: Right.

Steve: And so they kind of have, I don't know, I would say they have a responsibility, but there's something there.

MIKAH: Exactly. You're right. What do we call - that's a good - what do we end up calling, it's like, you walking along, and you see a baby bird that has fallen out of a nest. And it's not exactly that, I guess. It's not a great metaphor. But the point is, yeah, there are situations in life where you just happen to discover something. And if you know the impact that it can have, what is your responsibility?

Steve: Maybe the analogy is what obligation does a superhero have to come to the rescue of someone? It's like, well, if you can, don't you have to?

MIKAH: Yeah, that's a great way to put it, yeah. I like that.

Steve: Anyway, so they end up saying: "The proposed framework" - which, you know, they've structured here - "considers an organization's protective role, including protecting its own systems and customers, the time needed to act, and its ability to handle sensitive information. The same criteria would apply to comparable organizations, with maintainers deciding whom to involve ahead of disclosure.

"Each participant receives the information needed to carry out their protective work, from network logs with indicators of compromise to more sensitive working proof-of-concept exploits." So here they're talking about recognizing the need to sort of stovepipe as much as possible. I mean, containment is difficult, but crucial. And then they said: "Even limited details can reveal a vulnerability, so we weigh the risk of sharing them against the protection they enable.

"Teams following this framework would work with maintainers to give defenders time to prepare and reassess the timeline if there's active exploitation in the wild, a leak, a revealing public patch, or increased risk to users from stalled remediation." So again, being in real time writing this thing and reacting as necessary moment by moment.

And they finish by writing: "The Patch the Planet website brings together public disclosures and guidance to help operators identify affected systems, find the right update, and verify that the updated software is running properly. For PHP, it points to the maintainers' official patch and upgrade instructions and our partners' technical research. The guidance also covers interim mitigations, when to involve a provider or expert, and backup and recovery. Interim protections can reduce risk, but we'll make their limits and operational tradeoffs clear."

So the posting, that's about the first third. It goes on from there, becoming more generic and a bit more OpenAI-centric. And as I said, I'm unsure what role any given AI provider should really have. You know, they're providing the tool that was used to facilitate the discovery. And it seems to me that the discoverer of a massively impactful flaw needs to very quietly and carefully inform the publishers or maintainers of the system in question, and then those publishers or maintainers are best positioned to best know how to proceed. But it's cool that an organization like OpenAI that has government contacts and infinite resources, essentially, would then, you know, want to be able to, be willing to, and want to take an active role in helping this thing to, you know, go from private to fully public.

And as I said earlier, the industry really has a problem because there apparently are more of these. And we're just not as an industry set up for this. This has not been the way things have gone in the past. There have been a couple. I'll talk about them in a minute, like Heartbleed, which was a hair-on-fire, everyone running around "oh my god" event. But, you know, it stands out as something like that. Well, now we have this. And apparently there's more coming.

So, okay, so what happened? Let's take another look at this. The guys at VulnCheck checked in last Thursday about this event. They said: "Earlier this week" - and I should say they're also deeply involved. "Earlier this week PHP disclosed two critical vulnerabilities, CVE-2026-109618 and 109619." They're collectively known as "PHP2Shell." They wrote: "...that allow for unauthenticated remote code execution. VulnCheck researchers independently reproduced both vulnerabilities the night the news leaked. What have we seen?"

They said: "VulnCheck's own exploit achieves RCE on ASLR-enabled systems." Okay, acronym soup. RCE obviously remote code execution. ASLR is address space layout randomization. As we've spoken of before, and I think it was a couple weeks ago, Mikah, so not when you were listening, AI has successfully been employed in mitigating the protection offered by the randomization of where things are loaded in memory when an operating system boots. So ASLR, address space layout randomization, was a powerful mitigation implemented many years ago and that we've been relying on to make exploitation of vulnerabilities once discovered far less reliable and, in some cases, not worth pursuing.

Unfortunately, AI has now proven itself able to cut through that. So they developed, the VulnCheck guys immediately developed an exploit which achieves remote code execution on ASLR-enabled systems. They said: "Our Initial Access Intelligence team has a working RCE exploit for the CVEs on x86-64 with ASLR enabled on several PHP 8.4.x versions."

They said: "We see massive exposure." Okay, of course. "VulnCheck Target Intelligence found PHP on more than 2.6 million Internet-facing hosts, and 99.998% of them vulnerable or likely vulnerable. There is no fix for many." Now, this is a key that I have not seen mentioned a lot. They found it on 2.6 million Internet-facing hosts. Virtually 100%, 99.998% of them are vulnerable. And here it comes: "There is no fix for many. About 1.1 million of those 2.6, which is to say 42%, are running end-of-life PHP branches with no official patched release. And PHP," they write, "is often hidden. More than 469,000 WordPress and more than 124,000 Laravel hosts are running on PHP. Even if the system," they write, "doesn't advertise PHP, it can still be exposed."

And, they write: "Exploitation is now active in the wild: VulnCheck Canaries have recorded growing numbers of attempted PHP2Shell exploitation. We have observed attackers adopting at least five public exploit kits within a day, including one multi-stage attempt to bypass ASLR.

"There is also a flood of bad proof-of-concepts: Our research team has reviewed hundreds of different proof-of-concept repositories, most of which are fake or otherwise demonstrate artificial lab scenarios. VulnCheck Exploit and Vulnerability Intelligence data is being updated continuously as our team reviews new proof-of-concept code."

Okay. So we are seeing the familiar patching pattern play out here once again, but this time on a far more sizeable and dramatic scale. It is literally the case that AI enabled the discovery of a flaw that allowed most of the Internet's assets to be remotely compromised. That's huge.

Before we look at Matthew Prince's comments it's worth appreciating that this flaw had been known of for many years by the responsible people, but had been deliberately ignored because its exploitation had never been demonstrated. Now, think about that for a second, and ask yourself where else similar known but dormant problems might lie. Here's a piece of the timeline description posted by Calif. This is an author at Calif who writes in the first person.

So he says: "The year was 2004." Right? 22 years ago. "The year was 2004. The iPhone was yet to be invented, and our very own Stefan Esser was still the PHP security guy. Back then," this author writes, "I was a kid. I've followed Stefan's PHP research to improve my own understanding of PHP as an attack surface. Long before the 2008 call-time-pass-by-reference/interruption vulnerabilities and the more famous unserialize use-after-frees and POP(object) injections" - this is all, you know, inside baseball PHP security problems, but these are, like, this raft of problems PHP has had through the years - he writes: "Stefan unearthed the first PHP memory limit remote vulnerability in 2004." And I haven't specifically said so, but that's what this big bug was last week. It was a PHP memory limit exceeded mishandling bug. So the first one known of in 2004, 22 years ago.

This author says: "I remembered the vulnerability as it was described by Stefan as being critical enough that he would not release his exploit to the public. Eventually, two other commercial exploit vendors developed non-public exploits as well, CORE and Immunity Inc." In other words, they found it, and they sold it to bad guys. Other than those, we only had fake exploits circulating on IRC designed to compromise machines.

"The years went on, and all of the default unserializable classes with use-after-free or type confusion bugs in PHP were found one by one and patched as well. We verified this earlier in the year" - like this year - "earlier in the year when we released our first PHP research attempting to find any remnants that might have been missed. We found no exploitable vectors that would be exploitable by default.

"Then in 2020 came a report of a seemingly harmless bug, 78876, to the PHP project, noting that certain long variables could cause an out-of-memory condition and the failure to clean up temporary files. At the time, not much was made of the report. Fast-forward four years to 2024 when OSTIF and Quarkslab performed an audit of PHP and rediscovered the same OOM (out-of-memory) issue. However, exploitation was not proven, hence the bug was never fixed.

"Finally, we arrive in 2026. OpenAI invited us to join its Patch the Planet initiative, and PHP stood out as a high-impact target. We wanted the PHP Foundation involved, so Stefan connected us with Volker Dusch, who accepted our invitation. Then we got working hacking PHP.

"On August 1st, 2026, we made the discovery. We were not expecting to find anything critical. Everyone and every lab had already done extensive AI-assisted audits of this target. But only a few days after starting the audit and asking the right questions related to Stefan's earlier findings" - meaning prompting the AI with the right questions - "we managed to surface the first out-of-memory fatal flaw and the first of the three interruption points that allowed us to piece together a remote code execution against WordPress.

"Since the first bug isn't usually the best one, we continued looking for better variants. In the end, we found two more that lead to reliable exploits against mod_php and PHP-FPM." Which are basically PHP itself.

So what this means is that PHP is such a mess that even during a full security audit, fundamental architectural problems that do not directly and provably compromise PHP's security, but are nevertheless bad, they're just ignored because, you know, PHP is already such a mess. Had these well-known and long-tolerated out-of-memory mishandlings been addressed years ago, none of what happened last week would have transpired. As I noted, it's going to be interesting to see what the next surprises reveal and whether we develop a similar pattern.

So to that end, I want to next share a piece that appeared in The Atlantic last Friday October 23rd. That piece, written by Matteo Wong, who covers this area for The Atlantic, had the title "Welcome to the Valley of Pain." Okay. "Welcome to the Valley of Pain."

MIKAH: Oh, my goodness.

Steve: It had the subhead "An explosion of AI-powered hacks will reshape life online." Okay, now, that's not the subheading I would have given this piece since while I expect online life to be reshaped by AI and after all, that's already underway; right? I mean, it is being reshaped now. I doubt it will be directly reshaped by AI-aided malicious action. I mean, some, yeah. But, okay, we'll see. As I noted last week, behavioral change, that's more difficult, but it's probably what will be needed. The way we behave about security in organizations and even as individuals and in small companies. So I believe we're going to go through and survive a rough patch. We'll still be here on the other end, but we'll have more podcasts about, you know, what transpires in the meantime.

And, you know, actually surviving appears to be the feeling of the people that Matteo's article quoted last week. You know, as often is the case, the headlines of these things are chosen by editors, not by the people who write the column. So welcome to the Valley of Doom, or Pain.

MIKAH: Pain.

Steve: Okay, right. And here we are, an hour in. Let's hear again from Leo, and then I'm going to share what the Valley of Pain looks like.

Okay. So Matteo writes for The Atlantic on Friday: "This week could go down as the worst in the history of cybersecurity. On Wednesday, news emerged about two major vulnerabilities in PHP, a programming language used by 70% of websites, which prompted governments, hospitals, and companies around the world to take digital services offline as a precaution. There have already been millions of attempts to exploit these weaknesses.

"But the PHP vulnerabilities were just the most attention-grabbing in a series of alarming revelations. Security flaws in, or successful hacks of, several major software products and platforms - including some used by Netflix, PayPal, Walmart, and other big names - were announced the same day.

Yesterday, Quest Diagnostics, which runs lab tests for a third of American adults each year, disclosed that it, too, had been breached. When I asked Sam Rubin, a senior vice president at the cybersecurity firm Palo Alto Networks" - by the way, also a sponsor of the TWiT Network - "about the potential fallout from yet another hack, this one of the cloud platform Railway, which is used by Intuit and MGM Resorts, among others, he paused. 'I didn't see that one,' Rubin said. 'It's been quite a busy week.'

"Yet what's most alarming about the past few days is that by next year, this stretch of incidents, which some security experts are calling "Black October," may be remembered as normal, or even tame. Matthew Prince, the CEO of the Internet infrastructure and security company Cloudflare, told me: 'It's going to feel like, over and over and over again, we're finding more and more bugs in very critical pieces of software. This story, which feels unique and fresh right now - in a few months, it's going to feel like we get one of these every week.'" Oh, boy.

"Prince said" - this guy writes, Matteo is writing - "Prince said he knows of still more vulnerabilities that will be announced in the near future. For the next year or two, if not longer, the Internet is going to feel a bit touch and go, and more dangerous than we've come to expect. Matthew Prince said: 'It's going to feel like the sky is falling for a while.'"

Okay. And then Cloudflare's Matthew Prince echoes the primary thesis I've been voicing for the past few months, which our listeners will recognize. And it's the reason for this topic's title "Technical Debt Repayment." He said, or Matteo says: "There's a decades-long backlog of bugs in software written without much review or scrutiny, which was more or less fine so long as there was a very small number of skilled hackers. Over the past two decades, companies, governments, and individuals have been lulled into a false sense of security. Most people don't think twice about punching credit-card numbers into the website of a local restaurant or gift shop, or forking over a home address and phone number in just about any customer survey. But the Internet was always a dangerous place. We're just waking up from a trance."

Okay. This podcast's listeners have I think a better calibration on that. Yes, there are bugs. They haven't been a problem. They've been ignored. Or they, you know, they're only fixed when they're discovered, so they're latent, but unknown. AI is making them known.

Anyway, he goes on, writing: "AI agents can now be deployed as a cheap and near-infinite hacker army. In just the past couple of months, Rubin said, open-source models, which can be downloaded for free and used without restrictions, have become very good at finding vulnerabilities and automating attacks. In turn, the amount of malicious cyber activity has spiked. More websites will be defaced, more personal information will be stolen, and ransomware will proliferate. Rubin told me that the Internet is entering a 'valley of pain.' When I posed the concern that this is the new normal to Adam Meyers, a senior vice president at the cybersecurity company CrowdStrike, he told me, 'I think it's going to be getting a lot worse, actually.'

"Consider, for instance, that another set of vulnerabilities disclosed this week to the program Node.js. This software, which executes JavaScript code for dynamic web pages, is widely used by Fortune 500 companies, as well as in programs developed with AI. Meyers said: "These are the things the web runs on." Rubin said that because of how the platform is used, compromising just one software product that uses Node.js could infect tens of thousands of other organizations in what's known as a 'supply-chain attack.'

"And the skill threshold for launching cyberattacks has become almost absurdly low: CrowdStrike tied a spate of cyberattacks on South Korean banks late last month to a 26-year-old in China." And we talked about this a couple weeks ago. "Meyers said: 'After he got access to all the data, he turned around and asked AI, "How do I monetize this? Where can I sell this?"

MIKAH: Wow.

Steve: Yes. He didn't know what he was doing. "Meyers said CrowdStrike is detecting 'a new agentic adversary' - that is, an AI-powered hacking operation - 'a new agentic adversary. Every two hours a new one is appearing.'"

So this is what we've talked about before; right? The effect AI has of lowering the skill level required bar to the floor means that an entire new community of what were once just wannabe hackers and

attackers have now become empowered overnight.

The Atlantic's article continues, writing: "Navigating this will require rapid and tremendous coordination from basically every sector. For instance, the vulnerabilities in PHP were discovered months ago by a consortium of AI and cybersecurity firms. Then, Prince said: 'Very quietly, for the last several months, we and other industry partners worked to effectively put in place protections. Basically, temporary shields until the weaknesses could be fully addressed. But not everybody uses Cloudflare, and not everybody is quick to patch their software.'"

I'll just interrupt to note that web application firewalls, which is one of the services that Cloudflare offers, can be tweaked in order to prevent the types of queries that affect this software. So it was possible for Cloudflare because they stand in front of all the websites that are behind them to preemptively and quietly, without any disclosure, put a block in so that even until all of the sites behind them's PHP instances were updated, everybody would still be protected.

Matteo continues, writing: "Some of the cybersecurity experts I've spoken with are optimistic about AI. Eventually, they say [count me here], AI will be used to find and address many of the major cybersecurity flaws, and the Internet will become much safer overall. But before then, we may be stuck in a terrible limbo. Alex Stamos, the chief information-security officer at the AI company Cognition and a former chief security officer at Facebook, told me: 'This is going to last as long as it takes to rewrite and secure the billions of lines of code that have always had these vulnerabilities.'"

And Matteo finishes, writing: "And even as cybersecurity experts rush to leverage AI to save the Internet, the technology is introducing plenty of new problems." Okay. He writes: "All vibe-coded apps, for starters, are very likely to contain security flaws. And at a time when people need to be more vigilant about their digital hygiene, multiple cybersecurity experts brought up the widespread adoption of AI assistants with more than a hint of scorn. As Rubin put it: 'Our peers are using Muse and giving it carte blanche to use your credit card.'"

Okay. I'm going to interrupt here and just say, for the record, that I'm going to push back on that a bit. Not about Muse. Yeah, that's dumb. But about vibe-coding. We know that it is certainly possible for vibe-coding to produce insecure software. Many months ago we shared the story of a listener whose retired father asked AI to create him a website. "Make me a website." And the AI did what he asked. It created a website. Was it secure? Did it have any security enforcement? No. Because this guy's father hadn't known to, or thought to, ask for a "secure website." He literally just said "website."

MIKAH: Just make me a website.

Steve: And it gave him one. Yeah, here you go. Is it secure? No. So that's an extreme example, but it shows the trouble; right? There's no reason to believe that properly written software whose security is thoroughly tested and pounded on before its release will be any less secure than traditional software written by hand without automation. And I suspect that future software will incorporate far more security from the start. For one thing, we have much newer languages such as Rust which inherently resolve many of the problems that result from mistakes with earlier dangerous languages like C is the classic hyper dangerous language.

Human coders continue choosing C because it's what they know. It's what they grew up with. It's what their libraries, all the libraries they have are in C. But an AI, unlike a human coder, can simply be told to please use Rust. And it will. So I believe that in the longer term there's reason for optimism on the software security front. But we're not there yet. That's not today.

The Atlantic's article begins to wrap this up by writing: "The Internet is becoming a wilder place, and the path to secure it may require, in essence, a sort of pruning. Dan Guido, the CEO of the cybersecurity firm Trail of Bits, told me: 'You'll likely see more companies taking services off the Internet when they don't need to be there,' deprecating old websites and forms so that they cannot be targeted.

"Individuals might be encouraged to delete unused browser extensions, he added, so that there are 'fewer pieces of software that can access your data.' Alex Stamos said: 'People in general need to be more discerning of the companies they work with. You're just not going to be able to assume every site you use is secure.' That Pokmon trading-card website, the fantasy sports platform, or birding

platform you use could be hacked for a password you've reused elsewhere, or a credit card number, or worse. Other advice I heard, he wrote: 'Don't host your own email client instead of Gmail. Don't host your own file system at home instead of using the cloud. Google and Dropbox have the resources to defend against an onslaught of AI hacks, and you don't.'"

Okay. Wait a minute. Hold on. I would temper that a bit to say definitely do not host your own email client instead of using Gmail or your own file system if you do not fully understand the security implications of doing so. I assume when they talk about hosting your own file system, okay, they mean running a NAS. And we certainly know the danger of the temptation to have any management surface publicly exposed from that NAS. I mean, I've turned blue telling people no publicly exposed management interfaces.

I'm also somewhat annoyed by the idea that advice is being given to just turn everything over to cloud-based services. That's certainly not appropriate for much of this podcast's audience; but, sadly, I can see that it might generic sense for the sake of broader security overall.

The article finally concludes, writing: "If, eventually, we do find and shore up all the cybersecurity vulnerabilities out there, the result could be a more open web, where anybody can use widely available AI tools to run security checks at any time, and you'll feel safer sharing your personal information again. But another possibility is an even more centralized Internet, where only services and websites developed by large software providers are deemed trustworthy and reliable. The Internet needs new walls to defend against AI-powered attacks. But in turn, we might also feel walled in."

Well, okay. I would hate to have the Internet consolidated into a handful of mega-service providers. I don't think anybody wants that. But, you know, look at Cloudflare. I am more and more often encountering their "Verifying you're not a bot" interception page before I'm allowed to access a site which they are fronting for. Cloudflare offers a truly terrific service for web hosting, and we're seeing more and more of what was once the decentralized web moving behind their CDN and others. I think it's a sad thought, but that may be the shape of the future. I kind of hope not because that kind of centralization generally doesn't turn out well.

It's somewhat chilling to know that there are additional known problems that are still under wraps which will need to be dealt with in a fashion similar to what just happened with PHP. We're in for some interesting times, I think. In the case of PHP, the news of the flaw, as I said, and as we know, leaked prematurely, but thankfully only two days before the planned update announcement and release. So the PHP organization already, you know, they were just in the final throes of having all of their binary updates available and ready. So they were able to quickly advance their release by one day.

On one hand, they were mostly ready. But exciting secrets as we're discussed, Mikah, are difficult to keep. Here's a snippet from the PHP Foundation explaining from their viewpoint, from their perspective, some details of the collapse of the patch news embargo. They wrote: "The coordinated disclosure of these vulnerabilities worked well until two days ahead of the targeted release date.

"On Tuesday, October 20th, at around 15:00 UTC, the two CVE IDs began circulating on online forums and social media. They came with a wrong release time and a wrong CVSS score of 10.0. The actual score is 9.2," they wrote. "No technical details leaked. In response, the PHP project brought its pre-announcement forward and published it at 19:30 UTC. It also moved the release from October 22nd at 14:00 UTC to October 21st at 14:00 UTC." So advanced it by exactly one day. "Distributors were still packaging the fix, so releasing immediately would have left many users without an easy way to update." So basically what he's saying is that they turned the heat up to max in order to be ready one day early.

They said: "Moving the release forward by one day was the best compromise between acting quickly, giving people time to prepare, and making sure everyone had a simple update path when the vulnerabilities were disclosed."

They wrote: "As a result of the leak and pre-announcement around 20:00 UTC, Calif, one of the security firms that had reported the issues, mistakenly published a video. It showed their proof of concept working against a number of targets and revealed a few additional, though limited, details about the vulnerabilities." So, and that may be the company that you were thinking of, Mikah, Calif, as opposed, you know, because it's a [crosstalk].

MIKAH: Yeah, I think that's probably what happened, yeah, I misread.

Steve: And so these guys had spent two months not saying anything. But when - so but it's like the situation where you're worried when hostilities are escalating, and everybody is pointing their weapons at each other, that somebody's going to sneeze, and somebody on the other side is going to mistake that for a gunshot and then...

MIKAH: Yes.

Steve: So it's just like, you know, they mistakenly published a video. Even though they quickly deleted it, as PHP Foundation says: "Copies of the deleted video started circulating on social media soon after. The video contributed to accelerated development of various proofs of concept, which then increased circulation on October 21st.

"Mitigation advice we started providing to a few critical infrastructure providers around October 20th, 23:00 UTC, may have leaked. as well." So again, they began to feel they needed to get the word out. But these were less trusted partners. A few critical infrastructure providers around October 20th, 23:00 UTC, they said, may have leaked, as well. "The PHP security team closely monitored and inspected appearing proofs of concept after the leak, but did not identify any functional exploits at the time of public release and disclosure.

"On October 21st, 07:41 UTC, more than six hours before the coordinated release" - remember the coordinated release was at 14:00. So at 07:41, "Chainguard unilaterally," that is, okay, "unilaterally broke the embargo and published built images containing the fix. Like other distributors, Chainguard had received the patch in advance so its users could update the moment the vulnerabilities were disclosed. Any attacker could now study the fix in binary form while every other PHP user was still waiting for the release. This disregarded weeks of work by many parties toward a coordinated disclosure." Okay. So it's not...

MIKAH: It messes it up for everyone. It's incredibly selfish.

Steve: It's not difficult to imagine that Chainguard will not be offered early access to future critical release information, certainly not by the PHP Foundation, who are clearly annoyed by that, big-time. And I would imagine not by others who understand and respect the need for absolute pre-release information disclosure control. So Chainguard hurts themselves and their users, they're probably not going to be in the loop.

Okay. But say that they hadn't gone off half cocked, seven hours early. Imagine that everything had gone according to plan. The outcome would not have been that much different. No matter what the industry does, we've never had the capability of updating everything at once. So the moment PHP was announced as planned last Thursday, if everything had gone according to plan, rather than urgently a day earlier, most of the same dynamics would have ensued, right, come into play. Exploitation proofs of concept would have been created, and bad guys would have begun attacking with the goal of compromising PHP systems before they had the chance to get patched.

With 2.6 million systems out there identified, and 99.998% of them vulnerable, that's just too juicy a target to pass. So it's unclear to me how days of frenzied patching and attacking can be prevented. I just don't see it. It's going to be really - it'll be a function of what it is that needs to get fixed; right?

Of course now, as I mentioned before, the well-known exception to all of that is our web browsers. They're able to update immediately and more or less continually, and they do. This capability was universally developed because they represent the Internet's largest attack surface for their users, and because they've had so many problems in the past. Once that feature was added, it was self-continuing, self-perpetuating; right? But the software update model that most of the Internet is still using today has not changed in decades because it lacked the browser's level of urgency. And now it's too late. It's like, it's already out there. It's deployed. And it doesn't - there's no way to update it on the fly.

Now, in fairness, server infrastructure is not the same as end- user consumer browsers. I'm sure there are people listening who are going, hey, we can't have our critical infrastructure stuff updated on the fly. That just won't work. So it's not fair to suggest that the model of continuous background

autonomous updating could ever be applied to non-consumer enterprise or cloud environments. I get that.

One possibility that occurred to me is what I'll call "full stealth updating," where a horrible problem is quietly patched and resolved without the explosive fanfare and events that accompanied last week's PHP debacle. Now, of course, it's not practical everywhere. Like it's not practical for PHP. The idea would be that not until many months had elapsed and after most affected systems had been patched in the normal course of software maintenance would an "Oh, by the way, we fixed this horrible problem some time ago" be mentioned. Or perhaps, even better, nothing would ever be mentioned. I think the reason that the people who discover these sorts of big problems go on to hyper alert and, oh my god, we have to patch the planet is because they're afraid the bad guys will. It's like, if we were able to use AI to find this problem, we don't know that other bad guys won't.

So we have to fix it, and we have to raise an alert. And I love the idea of the stealth update that no one mentions, and it just goes out with regular scheduled repair, like out into the ether, until the point where it's no longer such a big problem. And then you mention it, and tell everybody who is running versions that are still older than any that were fixed, that it's time for them to get updated.

Anyway, unfortunately, Cloudflare's Matthew Prince was quoted in that Atlantic article saying: "It's going to feel like, over and over and over again, we're finding more and more bugs in very critical pieces of software. This story," he said "which feels unique and fresh right now, in a few months it's going to feel like we get one of these every week." So that sure doesn't sound as though whatever is going on, and that we don't know about, but he does, will be suitable for any sort of quiet stealthy updating. He's saying these fire drills are going to become somewhat routine. So it does sound as though we're in for a rough next few months, Mikah.

MIKAH: Yeah. And that makes me - that is the thing that I kept seeing is, here, how do I bring myself, there we go, that is something that I kept seeing across the board was people talking about hardening their own security. But more importantly, the people who are currently at these AI companies and the folks who communicate with them talking about how this is just the beginning; this is just the start. We have much more that we are looking at. I think someone was recently quoted as talking about, you know, right now we're looking at a PHP which already you kind of don't - some people don't think about because of how - you mentioned even in your notes today about it kind of being buried, or obfuscated.

But what happens when it's machine code that has flaws in it? What happens when it's down to firmware and then deeper layers? And how do we fix for that, and can you? And then just last week you talking about how a research version of Claude being able to just figure out how to move along a system and break apart and find the origin of the encryption, was it? It's all still very, like, ready.

Steve: It was Claude, a multi-agentic Claude-based attack where, you know, it filtrated some enterprise's network and was able to look around and take all the knowledge it had of everything on the Internet and inventory the network that it was now in and develop exploits on the fly.

MIKAH: They don't need sleep.

Steve: Wow.

MIKAH: They don't need sleep. They don't need anything, other than, you know, what they have, and especially when you do have the multi-agent functionality. Now, I wanted to ask you, and maybe you talk more about this, as well. But there have been some people who, since this first came out, have said that 70% number, it might be true, but it's not something that is that big of a deal, or isn't a concern because it won't touch most people due to things like Cloudflare being in the way, essentially. So how - do you think this is something that will end up being played on local news packages, essentially? Is it like Heartbleed was, where it had enough legs that people could kind of understand what was going on, and then that resulted in everybody talking about it?

Steve: Certainly when sites all over the Internet go dark because they need to disconnect in favor of being compromised. That affects people, and people see it and feel it. What I have continued to believe is that all the bad guys want is money. Hacking a company for the fun of it, that's lame now. Once upon a time that was a big deal. Now you're a lamer, if that's what you do. You want to

exfiltrate their data and then extort them in order to get money from them. That's the game.

And so I don't think we're going to see, I mean, the event has passed; right? I mean, PHP, that happened. I believe there will be systems which are forgotten, that are in the backroom, or as we often say on the podcast, in some dusty closet where they will not be updated. An attacker will be able to get in. It's also not the case, and we saw hints of this in some of the reporting that I shared, where the immediate proofs of concept are mature and robust and work regardless of PHP version.

There is still some time required to get this new weapon tuned up and polished and working correctly. So it will be used against PHP sites to allow bad guys to burrow in and then pivot from that server into the rest of the network. So I don't expect that, like, there will be a tidal wave of compromise or like some huge historic event. It's just that more ransomware will happen. More companies will be attacked. I received a beautiful piece of email earlier this morning. I'll share it next week. It was from a listener of ours whose small 30-person company was attacked by AI. And so it's like, not just big organizations, but any.

So my feeling is there isn't, like, any interest in creating an Internet worm because who cares? It doesn't make anybody money. What makes money is infiltrating an organization and arranging to blackmail them, extort them somehow. And there's never been anything more potent than this PHP nightmare to allow that to happen. I think we're going to be seeing trouble for quite a while downstream of this.

MIKAH: Do we think this is finally then the time - because there was something you said earlier that one of the researchers I believe was talking about. And I thought, this is something I said before we had generative AI going around and messing everything up, which was something along the lines of now you have to be worried that the Pokmon card site that you visit is going to get - we've been talking...

Steve: Right. That was - I agree, that was sort of a lame wrap-up to that column. I mean, I don't like to put my credit card into websites. I put it in - I have a PayPal, and I'm very happy when I see that a site will take PayPal because I want that, you know, an extra level of isolation. So, yeah, I don't think the character of the Internet is going to change that much. Most people are not affected by these bugs. Most, you know, users, end users, don't have PHP servers running out of their house. You know, they're not hosting their own web forums. So to me, this was a bad event, and it will be enterprises, not end users, who end up suffering. I don't think any of what's happening is going to change the character of the Internet at all for your typical end user.

MIKAH: Yet. Or ever, do you think? Because that's - I'm wondering if people will...

Steve: I don't think people really feel the Internet is a bad place. I mean...

MIKAH: That's fair. You're right,

Steve: Yeah. You know? I mean, it's like got social media. And it's got, you know, I mean, everybody...

MIKAH: It's got my Amazon.

Steve: People are buying things and roaming around. I mean, I don't think people think about it as being unsafe.

MIKAH: Yeah.

Steve: I think that enterprises are the real - are the entities that are concerned and should be concerned because they can be blackmailed.

MIKAH: Yeah, that's where the money is to be made, yeah.

Steve: That's where the money is.

MIKAH: That makes sense.

Steve: So how did this affect our listeners? Those who subscribe, as I mentioned, to receive the weekly Security Now! show notes, received a heads-up about this from me Wednesday morning. I called it a Flash Alert. And here are some of the replies that I received.

"Thank you, Steve. Spent the whole day in a war room today, patching." He said: "This reminds me of a few stories you mentioned previously, eerie to feel it this closely this time." And he then shared the blog posting from the PHP Foundation and cited three quotes from it: "The coordinated disclosure of these vulnerabilities worked well until two days ahead of the targeted release date. Also, one of the security firms that had reported the issues mistakenly published a video. And more than six hours before the coordinated release, Chainguard unilaterally broke the embargo and published built images containing the fix."

So he finished, writing: "Click farmers, clout chasers, accidental reveals, broken embargos. The security community truly has to deal with a lot these days. Well, at least it got a pretty page; right?" And then he points to the php2shell.com site. And he said: "Can't wait for your continued assessment of this and similar previous situations and how this might shape the future of responsible disclosure. Didn't think this is how I would write an email to you eventually. Greetings from Austria. Best regards, Lucas Seinfeld." And then he signed off saying, "Web software engineer for over 25 years and proud SpinRite owner."

I got also a short one that just said: "Thanks, Steve! Just got my business servers patched!" And another listener, Corey Austin, who described himself as "A Fellow Bipedal Carbon-Based Life-form" wrote: "Hi, Steve. I literally had been talking about this event with a customer as your alert email came in. Specifically, I said, 'Yep, this will be in the Security Now! podcast next week.' What synchronicity! Can't wait for next week's episode! Cheers, Corey."

Another listener replied: "Thanks for the notice. We had a hair-on-fire patching session this morning to get everything updated." And Jason Bache of NerdsLimited.com replied: "Thank you, Steve! You made the right call throwing this one wide." Then there was also: "Thanks, Steve! Bit of a shock to wake up to here in Australia." Dave Mitchell wrote: "Thanks, Steve. We're already all over it, and our clients are all over us! High-profile one, this. Signed, Dave."

And Andrew Johnson replied: "Thank you, Steve. I know you're usually hesitant about sending out-of-cycle emails, but this one's important." So Andrew's right. I am hesitant because I don't want to be a source of additional unwanted noise for anyone. Consequently, my out-of-cycle emails have been very few and far between. I think only once did I do one before. But I received nothing but strong encouragement in this instance, and it sounds as though there may be some cause for a few more of these surprise notes in coming months. It also sounded as though most of our listeners were already well aware of the event, but I did receive a note that made me glad I had not resisted sending this out.

One listener wrote: "Thank you, Steve!! You really saved our asses on this one. I truly appreciate this emergency warning, and I'm so glad I'm on your mailing list. I'm a long-term listener; I've been with you since Day 1, some 21 years ago. I'm also a SpinRite owner. I feel like in the age of AI (of which I'm deeply involved now) this kind of alert is going to become the new normal. Keep up the great work on the podcast! Your friend, Scott."

Another different sort of note was: "Thank you for that. I was hoping my little forum for a sports association (not search-engine indexed) was too old, but it's on PHP version 5.6." 5.4 and on being vulnerable. He said: "Oh well, I'll take a static archive and shut it down. It's not worth the effort to update!" And that was signed Alastair.

And then there was: "Hi, Steve. As I write this I'm in the trenches updating a client's super old (and unfortunately highly customized) PHP-based CRM system that's currently stuck on PHP 7.4 and MySQL 5.7. Had another client message me last week before all of this, asking, 'Why are they attacking us? It seems nonstop!' I explained, 'It's not you personally. It's just not you. They do it because they can.'"

And another listener of ours, Michel wrote: "Hi, Steve. Long-time Security Now! listener from Belgium. The PHP2Shell situation this week reminded me very much of something you've said many times about legacy code and forgotten components quietly increasing the attack surface.

"I run a small personal Plesk VPS (Virtual Private Server). When the PHP2Shell warning appeared, I immediately checked my only PHP website. It was already running the patched PHP 8.5.12, so at first sight everything looked fine. It wasn't. Plesk, itself, turned out to be running its own embedded PHP through sw-engine, which was still PHP 8.4.25 and vulnerable. Plesk 18.0.81 Update 4 eventually moved that to the patched PHP 8.4.27.

"While checking the server, I then discovered another nice example of accumulated attack surface: Plesk had about 25 PHP handlers available across several PHP generations, although my actual website needs exactly one: PHP 8.5 FPM.

"I started removing the unused versions. That exposed another hidden dependency: Roundcube webmail still requires PHP 8.3. So after cleaning things up I can now actually account for every remaining PHP runtime: 8.5.12 for my website. 8.3.36 for Roundcube. 8.4.27 Plesk's own sw-engine."

He said: "Everything else is gone or disabled. What struck me was that none of the individual decisions that created this situation themselves were particularly bad. Old PHP versions and handlers had simply accumulated as the server evolved. Until this week they were just 'stuff that's still there.' Until suddenly, every forgotten runtime became relevant. It felt like a very Security Now! example of a simple principle: Software you no longer need is not merely clutter. It's an attack surface you may eventually forget you even have. Thought this might be an interesting little real-world anecdote for the PHP2Shell discussion. Greetings from Belgium. Michel."

And yup. Things are a bit of a mess with PHP at the moment. Michel's experience is a perfect example of why it may be less easy to thoroughly clean up this PHP mess than is immediately apparent. It's not difficult to imagine bad guys who see a large potential payoff - and again, that's what it's about, it's about money - taking the time and trouble to determine where older versions of PHP might be hiding while still being exposed to the outside world. As I said before, I think this one is going to be with us for a long time.

Christy Ramsey shared a perfect example of what my Picture of the Week caption labeled "an artifact of LLM technology." Remember, as I said at the top of the show, what was described in that cartoon was not a bug. It was an artifact of the way large language models work.

So Christy Ramsey wrote: "Hi, Steve! Listening for years. We use a SpinRite Consultant license at the non-profit computer refurb shop where I work. It has its own computer with SATA power and data cables hanging out for drive attachments."

He said: "I use Claude to help write answers to questions from clients and for my radio show. It recently called me out" - meaning Claude recently called him out - "for saying 1,000 Microsoft Patch Tuesday updates a month." He said: "I pushed back, and here is its response." So Claude replied: "I was wrong, and I need to own that one visibly. I called '1,000 updates a month' a joke and said not to let it stand as a fact. It is a fact, and it is understated. Fact, confirmed from multiple sources: Microsoft's October 13, 2026 Patch Tuesday fixed a record-breaking 1,411 flaws, including one actively exploited and two publicly disclosed zero-day vulnerabilities.

"According to Microsoft, the latest updates resolve 1,413 CVEs, including 1,102 affecting Windows, 134 in Office, 61 in developer tools, and 36 in SharePoint Server. And it was not a one-off. The October update follows another unusually large Patch Tuesday in September, when Microsoft addressed more than 970 vulnerabilities."

Then it finishes: "Where my reasoning went wrong: I answered from memory. Patch Tuesday counts I knew ran roughly 60 to 180 a month, so four figures sounded like exaggeration, and I did not search before dismissing it. That is exactly the failure mode my own standing rule is meant to catch: verify before correcting someone. The lesson I am carrying forward is that a number that sounds absurd is a reason to search, not a reason to doubt the person who said it."

So then Christy finishes, writing: "I enjoy the show. It's an interesting time to be in computer support. Back when I started, we called Microsoft, and they charged per call. So we would have rotating techs keeping them on the line to get a server up. It was well over 12 hours. I left after 5:00 p.m., and the call was still going when I came back at 9:00 a.m. I didn't hang around the team working on it. They were getting grumpy. Then forums appeared, then websites and search, then online chat support, now AI. May you live in interesting times indeed." So anyway, that was a perfect

example of a state-of-the-art, I mean, this just happened, a state-of-the-art Claude screw-up where it disobeyed its own rule of searching before correcting someone.

So I just want to finish as I said I was going to, by sort of a summation of where I am, and it's only a couple paragraphs, about AI. Over in GRC's AI newsgroup we've been having some discussions about intelligence versus consciousness. A valuable participant there argues that we cannot claim that current LLM technology is not conscious. And we see a lot of people out on the Internet claiming that it is. I mean, even well-known scientists, AI researchers. He asked me: "What would it take to convince you an AI is conscious and not simulated consciousness? What would be the difference between simulated consciousness and real consciousness?"

Okay. So after giving that question some thought, I replied: "My threshold for that determination would be a demonstration of actual understanding of the subject matter. For me, the biggest 'tell' that we're all dealing with a powerful language model is that its occasional confidently-stated mistakes demonstrate that there is nothing behind the curtain that in any true sense knows what it is saying."

What has happened during the past three years, since the first emergence of ChatGPT and conversational large language models, is that they've grown larger, and those creating them have worked tirelessly to more deeply hide their inherent limitations. Without question (and with the brute-force expenditure of truly astonishing quantities of cash), LLM's use of language to simulate a sentient entity has dramatically improved. No question about it. But that's all that's happened. The simulation has been polished.

I believe that we will limp along using LLMs, because that's all we have for the moment, until we work out the right way to create a sentient machine. Large language models alone, no matter how large they become, are not it. Scaling them scales the quality of the simulation, but that cannot change the reality of its character. What an LLM is, is an incredibly tantalizing simulation of what it MAY be possible for mankind to create. And I believe we're likely to because so much money and livelihood has now been committed to this effort. Our ability to do so is not a foregone conclusion. But without the false promise of LLM technology we would today never have the scale of compute available, nor the required desperation, that might make it possible.

So that's really where I think we are. I think we have, which is not to say this is not useful, my god, I mean, I use Claude all the time. It is an incredibly useful facility. And it's going to get a lot of use. Is it sentient? Is it conscious? No. I see absolutely no reason to believe that it is. We keep seeing evidence that it's not, that it is an incredible simulation of creatures, humans, who are. And I hope we get one. I think there's a very good chance we're going to. But it's going to be a different approach.

MIKAH: This is one of those times where I do - I wish that Leo wasn't on vacation because I would be curious to hear his response to this. I do think that - the reason why is you and I are 1,000% aligned on that. I don't see this as the technology that is going to be the thing that gets us to sentience. And so I can't give anything to the listeners in terms of a disagreement between the two of us. But yeah, I don't know what - how Leo feels about that in particular. I would say probably feels the same way. But I don't know. Yeah.

Steve: Yeah. It is incredibly useful technology. It is amazing. But I know what it is. As I said last week, we know because we built it. We built a statistical language engine, and then were stunned when it started talking. You know, there wasn't a consciousness that started talking. It followed its statistics. And we put language in, and we got language out. And it turns out that, because a book is able to contain knowledge, language contains knowledge. So that's how this thing knows, you know, it is intelligent, but it is not conscious. It is not sentient.

So but what I really think is interesting is I think that such an insane amount of money has been poured into this that at some point I believe we're going to get it. And it's probably going to be these world models that take a different approach. It's not a large language model. It's a large reality model. And when we model reality, then I think that's when we begin to get something that actually understands what it's saying, instead of just sounding convincing.

MIKAH: And is that - then I have to wonder, I don't know that I'd want that to be the be-all, end-all because at that point you have to bring ethics into the situation in terms of tasking these things with

- basically at that point is it not slavery if you have them doing things for you? Whereas right now it's just statistics doing its magic.

Steve: Yes, and we are trying to imbue them with ethics so that they don't give answers that we don't want them to give. But that's a complete artificiality. All they are is a big knowledge base and then we are trying to then imprint behavior onto it. And what they're finding is the more powerful they become, the less well they take an imprint. And so they're like, the reason they're talking about pausing is like, uh, we need to figure out how to make this safe to go any further. I don't think they're going to. I don't think it inherently works.

And it is true that once we get a machine that models reality rather than a machine that models language, which is just a reflection of our reality, you know, the language we created to describe reality, once we have a network that models reality, we're going to have a whole 'nother set of problems. And yeah, if it's sentient, then don't we owe it something? Then, I mean, real problems begin to emerge.

MIKAH: Indeed they do. That's a whole - I don't know. Can the big - we were just talking on another show, I think it was on Windows Weekly, we were talking about a meteor. Can the meteor just come before we are victims? Anyway, sorry, go ahead.

Steve: Okay. So we're at two hours. Let's do our final break. And then I've got two topics that I want to finish with: this question of pulling the plug strategically, and what happened with an Indian certificate authority whose stunningly bad private keys used on their intermediate certificates were discovered, and how that authority is a certificate authority no more.

MIKAH: Okay. So before we leave this topic, which has rightfully dominated today's podcast, I want to make one final observation. One of the prominent responses to last Wednesday's PHP surprise was to temporarily pull the plug on PHP-based Internet services. This appeared to be mostly a European response, likely due to a neighbor effect, but it immediately reminded me of Russia.

It's never been entirely clear why Russia has seen fit to go to so much trouble to developing the means to quickly sever all external global Internet ties and switch to a totally internal network. But for several years past we've been covering their tests of that capability. It's a bit creepy since one motivation might be to prevent cyber-reprisals in the event that they do something that upsets the rest of the world. I'm hoping that it's just a defensive measure instead. But either way, the notion of pulling up stakes, disconnecting from anything and everything cyber, and hunkering down is an interesting strategy. And there's some history of this having been done in the past.

As we mentioned before, and you brought it up, Mikah, Heartbleed, the Heartbleed incident, where it was discovered that chunks of raw internal web server memory could be exfiltrated over a secure connection and proofs of concept managed to extract a running server's private TLS certificate and lots of other data. It turned out that wasn't all that could be extracted; right? I mean, basically whatever was in the server's memory.

In response to the urgency of Heartbleed, Canada's tax agency took its tax-filing website offline as a precautionary preventative step, and that was just three weeks before the national filing deadline for taxes, meaning that the timing was very inconvenient for them to take themselves down. The suspended tax services remained offline for four days, from April 9th through April 13th, and their filing deadline was then extended to May 5th in compensation. That was 12 years ago, back in 2014, when pretty much everyone's personal data was not already in the hands of the bad guys. As we know, there have been so many data breaches since that it's like, yeah, you know, I'm sure all my personal details are somewhere, so what are you going to do?

The fact that six hours before Canada's service suspension 900 Canadian Social Insurance Numbers had been "bled out" through the Heartbleed attack, through Canada's tax service, that was a bigger deal back then in 2014 than it would be now. It's like, yeah, okay, fine. Nevertheless, other Canadian federal departments were vulnerable too, and they also wound up being shut down two days later. The response to Heartbleed was yikes, disconnect. Pull the plug.

In the U.S. six years ago, in 2020, on a much smaller scale, we all remember the SolarWinds Orion mess which caused CISA to order all federal agencies to power down their vulnerable servers. Just turn them off. And this happened again the next year in 2021 with the Microsoft Exchange Server

ProxyLogon vulnerabilities. Better to just unplug when a known problem may be exploited before it can be patched.

Then we had the Log4Shell panic that swept the Internet world shortly before the end of the same year, 2021. In a reactive panic, Quebec shut down 3,992 government websites, including the education and higher-education ministry sites. The minister made the point that without an inventory, confirming which sites even used the Log4j library itself, just knowing would take days. So he compared it to checking every room in every government building to find out which one used 60-watt bulbs. It was like, we don't know. So pull the plug. Better safe than sorry. More broadly across Canada, the Canada Revenue Agency, the Employment and Social Development Agency, and Toronto's Metrolinx transit system all went offline out of an abundance of caution over the Log4Shell panic.

Also on a smaller, more local scale, the reaction to the Ivanti Connect Secure flaw that we had was CISA ordering every U.S. federal civilian agency to disconnect all Ivanti Connect Secure and Policy Secure VPN appliances within two days of the order. That directive hit 102 federal agencies which researchers believed should have probably been patched, it was a bit of an overreaction. But again, better safe than sorry, rather than disconnecting.

So my point is there is very well established clear precedent for implementing what I would call a strategic plug pull, if only for an hour or two. You don't have to be down for days. But when things look bad enough to warrant it, pull the plug. Last week, the cybersecurity agencies in Europe, Australia, Japan, and Britain all told their PHP users to either update immediately or, if for any reason that's not possible, take their sites offline until they're able to patch. Pull the plug until it's fixed. This contrasted with earlier pull-the-plug responses which were usually decisions made by individual governments or agencies.

In the Netherlands, many organizations last week followed the National Cyber Security Centre's example and pulled their own plugs, going dark, including the Port of Rotterdam, the government portal Overheid.nl, and all five of the country's largest political parties' sites, as well as several hospitals.

So I'm putting this strategic plug pull idea front and center because it's a strategy that might be worth keeping on hand and in mind. Although it can feel like overreacting, and perhaps it would be, depending upon the specific circumstances in the moment, it would likely be better than suffering an intrusion. We have no idea what Matthew Prince and others are aware of that we are not. We only know that, for a change, they are not saying "Something is likely to happen." You know, AI, ohhh, it's getting scary. No. This time they're saying "We have already discovered and know of several things that are definitely going to happen, and it's going to be like this PHP mess over and over and over." And they're saying, "But we're not yet ready to drop those bombs on the world."

So last Wednesday morning, when I learned of the PHP trouble, I immediately knew that GRC's forums were vulnerable. I also knew that whatever happened, my longstanding inherent mistrust of PHP would protect the rest of GRC, since as I noted earlier, that server's blast radius has always been deliberately contained. So I didn't panic. I first applied the quick PHP.INI configuration mitigations that had been provided, and then I installed the latest branch version and migrated all of GRC's PHP usage over to it. The same branch version migration meant that only the binaries, the PHP binaries needed to be updated. No other configuration changes were needed.

And realistically, compared to just about anyone else on the Internet, I'm about as low-value target as you could find, so I did not imagine that anyone would be coming after GRC immediately. So not panicking has its place. Also knowing that your PHP server can't hurt you no matter how badly it melts down, that's very valuable, too.

But that said, none of us have any idea what those next and now-anticipated surprises might turn out to be. You know, I will be quick to take my own advice to strategically pull the plug on GRC's Internet connection if any of what happens next turns out to warrant it. After playing dead on the battlefield to avoid being killed, Shakespeare's Falstaff famously said "Discretion is the better part of valor." That's the thing to keep in mind here. No one gets any points for remaining online in a cyber storm if that results in an intrusive compromise of one's servers and networks.

In these times of unprecedented cyber unrest, which I think is where we are, I promote the formalization of "The Strategic Plug Pull." No one likes an administrative service outage. The bad

guys will almost certainly be unable to attack anyone whose attack surface is offline. Right? I mean, remember that. That's how you sell this. And consider that any event that would be large enough to probably cause you to pull the plug to take your or your enterprise's services offline would obviously be big enough news that you would not be alone, or at least you would be understood.

I'm not a doomsayer, but we're surrounded by those in the know who are saying that there will be some rough times ahead. Perhaps we won't be affected. We won't know until we do know. But as I said, no one can attack an attack surface that's offline. So finally, and this is the point of this, if a bit of preparation for the possibility of needing to strategically pull the plug would make that decision easier and less painful, make doing it easier and less painful, now might be the time to think about it and to prepare for that possibility. Everyone's situation, responsibilities, and reporting structures are different. I get that. Maybe it's just not possible. But perhaps giving this idea some thought would be good because, again, apparently that's not an if. It's a when. And it's a series of significant whens.

Okay. Last topic, this eMudhra mess. We need to talk about the eMudhra mess which broke last Tuesday, the day before the PHP debacle. It was so utterly insane that I would have normally gleefully pounced on it as this podcast's topic of the week, if even it had not been overshadowed by PHP, which it was. But, boy, is this a fun one. Well, fun because I don't think anybody got hurt. But wow.

The private keys used at the time, last Tuesday, by the universally trusted intermediate certificates used to sign the TLS and code signing certificates issued by eMudhra, a Certificate Authority based in India, have not only leaked, that is, the private keys of the intermediate certificates, which would allow false signing under those certificates' trust, they had not only leaked. But those private signing keys had truly inexplicable values. This created a scandal so big that there was no recovery from it. Consequently, eMudhra is no longer in the certificate business. Gone, dead and gone overnight.

I'm going to let Ars Technica's Dan Goodin fill in the details. He wrote: "Security personnel around the world are on high alert Tuesday following the discovery of eight TLS certificates with private keys so weak they could allow anyone to mint trusted browser and app credentials for any site on the Internet." And I'll add also, sign any code. But Dan's 100% correct about the impact of this.

He continues: "As of late morning Tuesday, browser makers and some major apps and programming languages had blocked all eight of the intermediate certificates. There are no reports of certificate transparency logs, which track the issuance of all browser-trusted certificates, showing the compromised keys being used maliciously. Meaning this was found before it could be abused by bad guys. The threat remains far from over, however, because it will be hard and time-consuming to update other key parts of the Internet, such as Linux distributions, embedded and Internet of Things devices." And I'll get back to a lot of that here after I'm through sharing what Dan wrote. One word he wrote: Bonkers.

"The eight intermediate certificates belong to India-based certificate authority eMudhra Limited. An informal advisory published to emudhra.we-have-a-problem.com on Tuesday posted the message..."

MIKAH: Love that.

Steve: Yeah. "We have a problem here - Wayne." Those words were digitally signed using the private key of one of the eMudhra certificates. In other words, it was proof that anybody could sign anything. "Armed with proof that the reporting party had control of the key, defenders scrambled because the same key could be used to sign unauthorized certificates that would be trusted by millions" - like hundreds of millions - "of server clients worldwide." Basically, everything would trust anything that that key signed.

"Filippo Valsorda, a cryptography engineer and cryptography maintainer for the open source Go programming language, told Ars: 'This will be a massive cleanup job, bigger than Heartbleed. There will be a very, very long tail - IoT devices, Docker containers, forgotten Linux machines, unmaintained CLI tools that shipped their own copy of the roots (which is common in some ecosystems, like Python and Java). Also, Linux distributions are still rolling out updates. Some of them are not yet ready, and then system admins will need to apply them to a lot of machines.'"

Okay. So I'm going to pause again to point out that, as we're going to see, the truly inexplicable nature of the discovered private keys that eMudhra's own root certificate signed, and that Sectigo

also cross-signed, means that any device that carries its own internal root certificate store without some provision for that store being updated which means anything with firmware that needs to generally connect to Internet resources will now and forever, probably, fully trust any certificate it might receive from anything and anyone that's signed by one of those now utterly compromised intermediate certificates. It really is a mess. It is a horrible problem. It's not going away. But wait, there's more.

Dan's reporting continues: "Heartbleed is the name of a critical vulnerability in the OpenSSL cryptography library. In 2014, it came under mass exploitation. Exploits let many unaffiliated hackers decrypt TLS-protected exchanges between parties. They also recovered private TLS certificate keys, session cookies, and passwords. The attacks compromised thousands of servers, some belonging to large companies and government agencies around the world.

"Recovery required a long series of steps that took years. Steps included patching OpenSSL on all affected devices, reissuing millions of possibly compromised TLS keys, revoking old certificates, rotating passwords and session tokens, and updating embedded devices and other forgotten systems - something that was extremely time- and resource-intensive, especially at Internet scale.

"Valsorda explained: 'Every time you need a lot of different people in different contexts to do something, it's going to take a long time. This time it's worse because there are more contexts, more people, and more heterogeneity.'"

Okay. And here comes my favorite part of what Dan phrases poetically. He wrote: "The compromised eMudhra keys were defective at a level that's challenging the imagination of many of the engineers studying the incident." Okay. Once again, this is just beautiful. "The compromised eMudhra keys were defective at a level that's challenging the imagination of many of the engineers studying the incident." Then he explains. "A normal NIST P-384 elliptic curve key, an ECC private key, contains roughly 384 bits of entropy" - that's where the 384 comes from, 384 bits - "a size allowing almost 2^384 power, which is a number, if expressed in decimal, with 116 digits." Right?

So we've seen these things; right? Just looks like gibberish, 116 decimal digits is how big this key would be. "The private key sizes of the compromised keys" - in other words, the private keys that were found to have been used by eMudhra's intermediate certificates - "the private key sizes of the compromised keys ranged from 3 to 8 bits." Not 384 bits.

MIKAH: Three to eight bits.

Steve: Between three and eight bits. "Astonishingly, the private keys themselves, the actual private keys, were 4, 56, 66, 115, 155, 230, and 237." The actual key values.

MIKAH: How does this go for as long as it's gone, even?

Steve: Again, yes. And who knows what else is out there? Okay. So because this is utterly unimaginable in this day and age, I want to make sure everyone understands what Dan just correctly described; right? We're all used to seeing a typical cryptographic key; right? It's some ridiculously long jumble of decimal or hex digits that would be impossible to remember, let alone almost impossible to even correctly write down without making a mistake somewhere. In the case of a standard NIST P-384 bit key, that key would be 116 decimal digits long.

So against that backdrop, one of the private keys that the entire world was trusting until last Tuesday, and much of the non-updatable embedded world will continue to trust forever, was the single digit number "4." Not some jumble of 116 digits. Just "4." And that others, as Dan noted, 56, 66, 115, 155, 156, 230 and 237. We probably already have memorized a few of those, just ourselves. You can't memorize a 116-digit number. Well, people do memorize pi, so it guess it's possible. Or like the front of pi. It's true that a private key can be anything, and that a private key of "4" is every bit as valid as one having 116 digits, rather than just one digit or one bit. But something must have gone very, very wrong somewhere for this to have happened.

Dan's story continues, explaining: "Valdorda said: 'That is extremely in the "this should not be possible" category. Maybe some horribly dangerous key generation interface has a "size" parameter, and they set it to 1 by mistake?' Ben Cartwright-Cox, director of BGP.tools, said: 'My guess is that it's

example code that accidentally leaked into production, but it's hard to know. But it's bonkers that no one noticed.'"

Dan wrote: "At this early stage, the most immediate threat appears to be for those using VPNs, especially in countries with governments likely to maliciously intercept traffic. There are likely many other cases of concern. End users largely have no recourse other than to hold off using many non-browser forms of TLS communication until they receive assurances that updates are in place." Right? Because it's only browsers that are, like, Johnny-on-the-spot able to instantly update themselves. And they all did. But other TLS communications, any encrypted comms that use standard TLS and certificate hierarchies, are in danger.

"As noted," he writes, "most browsers have already blocked the intermediate certificates. The exception appears to be Edge running on Windows 10, which at the time this article went live was still connecting to 12 vulnerable host sites. In response to questions asking why and if the company has plans to fix it, a Microsoft representative said the company will be in touch if it has anything to 'share.'" Okay. "Meanwhile," he writes, "Signal, WhatsApp, Wireguard, and other apps that do not rely on TLS for confidentiality were never affected. The eMudhra.we-have-a-problem.com site show apps known to be patched so far." And I have a much better link that I'll give everybody at the end.

"The incident is the latest to expose a fatal flaw in the TLS regimen. It's only as strong as its weakest link, which," he says, "in this case was eMudhra. Fortunately, transparency logs account for much of that failure, but this measure requires parties to actively monitor massive feeds and works only after unauthorized certificates have been issued. eMudhra representatives were not immediately available to answer questions."

Dan then finishes his piece by somewhat understating the consequences, I think. But I'll explain. He writes: "One caveat: Exploiting compromised certificates requires an active adversary-in-the-middle attack, meaning the attacker must act as a proxy between two TLS parties" - that's what's not quite right - "and receive encrypted traffic, decrypt it, reencrypt it, and send it to the other party. Such capabilities, in some cases, are usually limited to only the most well-resourced attackers. People with control of a local network at a coffee shop, university, or other organization face much less of a challenge."

Okay, well, what Dan described is true as far as it goes. But on-the-fly traffic interception, which is what he's talking about, is only one of many ways of abusing a compromised certificate. The gift that the bad guys have been given with this eMudhra debacle is the ability to create their own fully trusted certificate for any TLS or Code Signing use. This means that they don't need to perform a complex traffic intercepting man-in-the-middle attack. They can much more easily impersonate any website and server on the Internet. They can create a fake site for any domain they wish. The good news today is that no one's web browsers will fall for that anymore. Not with those certificates. But the bad news is that there are many other instances where, as those Dan quoted for his piece, said that the static trust carried inside embedded and IoT devices may be subject to abuse lasting those devices' entire lifetime.

And as I said, I've included by far the most useful link I have found for answering any additional questions about the current state of this mess. The site's creator and owner, a Brian Vitko, notes at the bottom of the page in question, he said: "Built by someone who actually had to push Chrome to 5,000 endpoints on a Tuesday." Meaning last Tuesday. Anyway, the site is getrff.com. You can check out his blog, and he's got a blog link in his menu at the top. And you're looking for - he's blogged many times since last Tuesday, and this is a cool site. I recommend it to all of our listeners. Take a browse around there. The blog post is titled "eMudhra Compromise Linux Servers."

So anyway, so again, just GetRFF, getrff.com. He makes a very nice-looking Windows remote management tool set which our listeners might also find interesting. And having poked around the site a bit, I like this guy's site a lot. I'm glad this led us to him. So check it out.

Anyway, we survived PHP. We survived, you know, it wasn't earth-shattering. It was earth-shaking. My takeaway is the problems with PHP are going to be those in corners of the Internet that are not patched, yet are sitting on networks which bad guys can then pivot to once they compromise PHP. It takes longer than a few hours to develop a robust exploit. I'm sure that's where a lot of time has gone now in this past week. Companies are going to get compromised. They're going to get attacked. They're going to get infiltrated. They're going to get their data stolen, and then they're

going to get blackmailed and extorted. That's the first of what those in the know say we have in store for us. So buckle up.

MIKAH: Buckle up and...

Steve: Buckle up and get prepared to pull the plug.

MIKAH: And look. If you see my dental records online because my dentist used an old server that was pre-PHP...

Steve: Well, then we're going to know why your teeth are so white.

MIKAH: Don't look at it, okay? Stop looking at my dental records. Yeah, I guess this is my last little question for you as you talked about really this is a problem for the enterprise. But it does end up becoming a problem for the end user when it comes to data, personal data making its way out there. Do you have any tips that you'd like to - when someone comes to you who's a normo, and says, "I know you know a thing or two about security. What should I be doing?"

Steve: It's freeze all your credit. That's the number one thing that everyone should do. And we've been - I've been, you now, touting this now for decades, is lock down your credit. Most, you know, a young person like yourself, you may be buying things, you know, cars and who knows what? So you do need to have a credit report available in order for someone you want credit from to issue you credit. I mean, even now, like renting an apartment you often have to pass a credit check.

MIKAH: Yeah, you do, mm-hmm.

Steve: But thanks to this being a problem, last time I needed to do that, in fact I bought a home, I think that's what it was. Oh, no, I did finally a couple years ago buy a car. I like to, you know, I drive mine right into the ground.

MIKAH: Same.

Steve: So the one I had was 20 years old. But still...

MIKAH: That's what I do, too. I drive mine into the ground.

Steve: What I discovered is you are able to put an automatic refreeze on your temporarily unfrozen. So I asked the company I was buying from which bureau do you use, found out what it was, went to them. I had a longstanding perpetual freeze, and I told them to unfreeze it for a few days in order to pass the credit check. So anyway, that's really - that's the only thing you can do. In California, we do have some strong privacy protections. There is a statewide requirement. You can go to the site, I've forgotten what it is now, go to the site privacy.gov, maybe? Anyway, where you're able to - California will enforce all registered data brokers to delete you if you compel them to. So we have that in California. Maybe that will spread.

MIKAH: Fingers crossed.

Steve: What I tell everybody is just lock, you know, and not lock. A credit lock, it turns out, freeze and lock are different things. Locking sounds like what you want. No. You want a freeze. You want to freeze your credit. And that's the only thing you can do because our private information is all out on the Internet.

MIKAH: Absolutely.

Steve: And boy, identity theft is a nightmare that nobody wants to go through. Proving to somebody that you did not apply for that credit card, run it up, and then refuse to pay. It's like, no, it wasn't me. Well, you know...

MIKAH: Prove it.

Steve: Somebody who it was would try to claim that in order to get off, you know, to dump the balance of stuff that they actually did purchase. So it's a mess.

MIKAH: Yeah. I want nothing to do with that. Well, that does bring us to the end of this episode of Security Now!.

Steve: And hopefully a quieter podcast ready for us next week.

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/