PQ3

Description: Last week we covered a large amount of security news; this week, not so much. There are security stories I'll be catching us up with next week. But after sharing a wonderful piece of writing about the fate of Voyager 1, news of an attractive new Humble Bundle, a tip of the week from a listener, a bit of SpinRite news, and a number of interesting discussions resulting from feedback from our listeners, our promised coverage of Apple's new "PQ3" post-quantum safe iMessage protocol consumed the entire balance of this week's podcast budget, bulging today's show notes to a corpulent 21 pages. I think everyone's going to have a good time.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-964.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-964-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with lots of security news, including a look at Apple's new PQ3 encryption. Seems Apple might have oversold its capabilities.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 964, Episode 964, recorded Tuesday, March 5th, 2024: PQ3.

It's time for Security Now!. We are not for sale, but we are for free here for you right now. Steve Gibson, our man in charge.

Steve Gibson: Actually, Leo, for the right price, I have a feeling we would be for sale.

Leo: Yeah, I mean, everybody has a number. I'm not saying that. Yeah, so I'll take that sign off, yeah.

Steve: That's right. No.

Leo: Hi, Steve. Good to see you.

Steve: Great to be back for your pre-vacation podcast.

Leo: Ah, yes. I'll be gone for, I should tell everybody, I'll be gone for two weeks. Mikah will be filling in, yeah.

Steve: Thought we'd catch you before you leave. So this is, as I was wondering with you before we began recording, whether this was the shortest title we've ever had for a Security Now! podcast. I mean, maybe. But I guess there could be one called IQ or something in the past.

Leo: I think this is the shortest. Pawn to Queen 3 is the name of this show.

Steve: That's right. Or Post-Quantum Level 3.

Leo: Oh, what's that? Ooh.

Steve: Well, exactly. That's something that Apple just made up out of whole cloth because...

Leo: Of course they did.

Steve: They said, we want to be special, and we're going to denigrate all of the other things that have come before us that are down at Level - they even have Level 0, where I think they miscategorized Telegram, saying that it didn't have end-to-end encryption. It's like, what are you talking about? Anyway, I'm a little annoyed with Apple...

Leo: Uh-oh. Uh-oh.

Steve: ...after learning about this. But speaking of the podcast, before I forget, I brought whole site search back up at GRC. We've had a number of our listeners who have written saying, hey, I want to, you know, you say you have transcripts, and I know you do, but I can't find what I want. And of course, yes, you could use Google and put site:grc and then whatever it is. Anyway, and in the upper right of all of the pages of GRC has been a search box for a long time. But Google changed something that broke it. And I, you know, I figured, well, I've got to get SpinRite done. So anyway, SpinRite is done, so search is back. So I'm just saying that the whole site search, and you can narrow the search to just Security Now! podcast transcript search, if you like. So that's there.

We've got a fun podcast. Last week we covered a large amount of security news; this week, not so much. There are security stories which I'll be catching us up with next week. But after sharing a wonderful piece of writing about the fate of Voyager 1, news of an attractive new Humble Bundle, a tip of the week from a listener, a little bit of SpinRite news, and a number of interesting discussions resulting from feedback from our listeners, our promised coverage of Apple's new "PQ3" post-quantum safe iMessaging protocol consumed the entire balance of this week's podcast budget, bulging today's show notes to a corpulent 21 pages.

Leo: Whoa.

Steve: So we've got lots to talk about, and I think everyone's going to have a good time. And of course we've got a great Picture of the Week.

Leo: That would have been a good title, though. Corpulent. You might have considered...

Steve: Corpulent, a Corpulent Podcast, that's right.

Leo: You might have considered that. I have the Picture of the Week teed up. I have not yet looked at it.

Steve: One you will appreciate.

Leo: Oh, nuts. My screen has changed again. Why don't you describe it while I push the buttons here.

Steve: Okay. Well, so we have one of the wonderfully classic fake O'Reilly...

Leo: Oh, I love these, yeah.

Steve: ...programming coding books. You know, and this one is titled "Coding..."

Leo: So true.

Steve: "Coding with GPT." And then the subtitle is "Introducing the uncanny valley into your codebase." Because, you know, as we would say on this podcast, "What could possibly go wrong?" And then, you know, O'Reilly is wonderful because they've always got some animal of some sort, I don't know how the animals are chosen, if there's any rhyme nor reason, or if they just roll the dice.

Anyway, this one, of course, in keeping with GPT, has a completely bizarre fictitious made-up animal. It's got the head of a dog with cute little floppy ears looking off to the left, but then for its body it's got like the backend of a goose or a bird, a turkey, who knows what it is, of some kind, whose feet are on backwards. And then at the very end it's got like a tail of some other sort of bird looking, I mean, it's just a weird, bizarre animal. And I don't remember where it was, Leo, but it was on one of your podcasts, I think it might have been on a MacBreak Weekly a couple weeks ago, where we saw some weird animation of like a bunch of monkeys that were...

Leo: From Sora, yeah.

Steve: It was just, it, like, hurt you to watch it because they just kind of kept turning around and merging in...

Leo: Hallucinating, yeah.

Steve: Oh, wow.

Leo: I remember.

Steve: Anyway, so this is a hallucinated animal, and I would recommend that everyone keep ChatGPT away from their code. Actually, we'll probably get to it next week. One of the things that I didn't have room for this week, Ben Nassi, a person whose work we've covered before, he was at the University of the Negev, they've produced a ChatGPT worm which they call Morris II.

Okay. So we've had some fun keeping an eye on Voyager. And on the occasion that we may have finally lost control of this intrepid explorer, I found a wonderful piece of writing about this that I know our listeners will enjoy and appreciate. It's a blog post by someone named Doug Muir titled "Death, Lonely Death." And Doug writes: "Billions of miles away, at the edge of the Solar System, Voyager 1 has gone mad and has begun to die.

"Let's start with the 'billions of miles.' Voyager 1 was launched in early September 1977. Jimmy Carter was a hopeful new President. Yugoslavia and the USSR were going concerns, as were American Motors, Pan Am, F.W. Woolworth, Fotomat booths, Borders bookshops, and Pier 1. Americans were watching 'Happy Days,' 'M*A*S*H,' and 'Charlie's Angels' on television; their British cousins were watching 'George and Mildred,' 'The Goodies,' and Tom Baker as the Fourth Doctor.

"If you turned on the radio, 'Hotel California' by The Eagles was alternating with 'Dancing Queen' by Abba (and, if we want to be completely honest, 'Car Wash' by Rose Royce). Most cars still ran on leaded gasoline, most phones were still rotary dial, and the Internet was a wonky idea that was still a few weeks from a working prototype. 'The Thorn Birds' was on top of everyone's bestseller list." Blah blah blah. He goes on to sort of set the place for us, reminding us also that "It was the summer of 'Star Wars.' And Voyager 1 was blasting off for a tour of the Solar System.

"There's no way to pack the whole story of Voyager 1 into a single blog post. Here's the TLDR: Voyager was the second spacecraft to fly past Jupiter, and the first to take close-up photos of Jupiter's moons. It flew on past Saturn, and examined Saturn's moon Titan, the only moon with an atmosphere. And then it flew onwards, on and on, for another 40 years. It officially left the Solar System and entered interstellar space in 2012. It just kept going, further and further into the infinite emptiness." And he says: "You know about the Golden Record? Come on, everybody knows about the Golden Record. It's kind of a hokey and cheesy idea; it's also kind of amazing and great."

He says: "Voyager has grown old. It was never designed for this. Its original mission was supposed to last a bit over three years. Voyager has turned out to be much tougher than anyone ever imagined, but time gets us all. Its power source is a generator full of radioactive isotopes, and those are gradually decaying into inert lead. Year by year, the energy declines. The power levels relentlessly fall. Year by year, NASA has been switching off Voyager's instruments to conserve that dwindling flicker. They turned off its internal heater a few years ago, and they thought that might be the end. But those 1970s engineers built to last, and the circuitry and the valves kept working even as the temperature dropped down, down, colder than dry ice, colder than liquid nitrogen, falling towards absolute zero.

"Voyager stored its internal data on a digital tape recorder. Yes, a tape recorder, storing information on magnetic tape. It wasn't designed to function at a hundred degrees below zero. It wasn't designed to work for decades, winding, rewinding, endlessly re-writing

data. But it did. Voyager kept going, and kept going, until it was over 15 billion kilometers away. At the speed of light, the Moon is 1.5 seconds away. The Sun is about 8 light minutes away. Voyager is 22 hours away. Send a radio signal to it at lunch on Monday, and you'll get a response back Wednesday morning." He says: "I could go on at great length about Voyager the discoveries it's made, the Deep Space Network that has maintained contact with it over the decades, the ever-shrinking crew of aging technicians keeping it alive on a shoestring budget, how amazing has it all been.

"In 1990, just before Voyager's camera shut down forever, the probe turned around and looked backwards. It zoomed in and took a picture of Earth. By that time, it was so far away that Earth was just a single pale blue pixel. Seeing the blue pixel, Carl Sagan wrote: 'That's here. That's home. That's us. On it, everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every 'superstar,' every 'supreme leader,' every saint and sinner in the history of our species lived there - on a mote of dust suspended in a sunbeam.'"

He says: "Voyager kept going for another 34 years after that photo. And it's still going. It has left the grip of the Sun's gravity, so it's going to fall outward forever. Here's a bit of trivia: Voyager 1 currently holds the record for most distant active spacecraft. It's not even close. The only other contender is Voyager's little sister, Voyager 2, which had a different mission profile and so lags billions of kilometers behind its older sibling. Here's another bit of trivia: If you're reading this in 2024, it's very unlikely that you will live to see that record broken. There are only two other spacecraft outside the Solar System, Voyager 2 and New Horizons. Both of them are going to die before they get as far as Voyager 1. And nobody not NASA, not the Chinese, not the EU is currently planning to launch another spacecraft to those distances. In theory we could. In practice, we have other priorities.

"We thought we knew how Voyager would end. The power would gradually, inevitably, run down. The instruments would shut off, one by one. The signal would get fainter. Eventually either the last instrument would fail for lack of power, or the signal would be lost. We did not expect that it would go mad. In December 2023, Voyager started sending back gibberish instead of data. A software glitch, though perhaps caused by an underlying hardware problem; a cosmic ray strike; or a side effect of the low temperatures; or just aging equipment randomly causing some bits to flip.

"The problem was, the gibberish was coming from the flight direction software, something like an operating system. And no copy of that operating system remained in existence on Earth. This is a problem NASA long since solved. These days, every space probe that launches leaves a perfect duplicate back on Earth. Remember in 'The Martian' how they had another copy of Pathfinder sitting under a tarp in a warehouse? That's accurate. It's been standard practice for 30 years. But back in 1977, no one had thought of that yet.

"Voyager Mission Control used to be a couple of big rooms full of busy people, computers, giant screens. Now it's a single room in a small building in the San Gabriel Valley, in between a dog training school and a McDonald's. The Mission Control team is a handful of people, none of them young, several well past retirement age. And they're trying to fix the problem. But right now, it does not look good. You cannot just download a new OS from 15 billion kilometers away. They would have to figure out the problem, figure out if a workaround was possible, and then apply it, all with a round-trip time of 45 hours for every communication with a probe that is flying away from us at a million miles a day. They're trying, but nobody likes the odds.

"So at some point not tomorrow, not next week, but at some point in the next few months they'll probably have to admit defeat. And then they'll declare Voyager 1 officially over, dead and done, the end of a long song."

Leo: If you want to know more about this, there's a wonderful documentary, came out last year. I'm playing the trailer from it.

Steve: Good.

Leo: "It's Quieter in the Twilight." And it focuses on those people, working in that little office in San Gabriel Valley, who have gotten gray with Voyager. There it is. There's the little office. It's a great story. In fact, during the filming of the documentary, that's when they had decided they had to turn off the heater. So it's quite a dramatic moment when they're making that decision and what the cost of that will be and so forth. These are the people, the handful of people still running the Voyager mission. And they've gotten a lot farther than anybody thought they would, which is why they're getting kind of long in the tooth.

Steve: That looks like Gramps.

Leo: It is, it's great. This is a fabulous documentary. I couldn't recommend it more highly. It's called "It's Quieter in the Twilight." And they're not just talking about Voyager. I think they're talking about the scientists, the engineers, the mission specialists that made this happen. And they're still running this distant, distant object. It's kind of cool.

Steve: It really is something to be said, Leo, for our ability to create something with this kind of endurance in that kind of environment.

Leo: No kidding.

Steve: I mean, it is hostile out there. You know?

Leo: Yeah, yeah. Well, it feels like it's just luck. But on the other hand, NASA keeps doing it. Remember...

Steve: Yeah, the same thing with Rover.

Leo: Exactly.

Steve: What, Spirit and, I want to say Accountability, that wasn't.

Leo: Yeah. They finally had to end the drone, the little hovercraft, single rotor hovercraft they had that was flying around. What a story. But they only expected to

do a few, a handful of flights with that, and they got dozens. It's just - it's very impressive, I agree. It really is.

Steve: Yeah.

Leo: And it's a great story. And, hey, salute to Voyager.

Steve: Yeah. I have a feeling that I will briefly let everyone know when they finally pull the plug. But otherwise...

Leo: Watch the documentary.

Steve: ...we've enjoyed - yeah.

Leo: So good, yeah.

Steve: Good. So I got a very cool tip of the week from Patrick Johnson, who wrote: "Hi, Steve. Long-time listener, and when downloading 6.1 I discovered that I've been a SpinRite owner for 10 years as of Tuesday. I was surprised to hear in SN-963 that Firefox had let you keep the dedicated search box for so long. I forgot when it was turned off by default for new installs."

Leo: That was my reaction, too.

Steve: Right, yeah. "But I did keep turning it back on for quite a while until I discovered that the CTRL+K shortcut for search still worked."

Leo: Hmm.

Steve: Yeah, huh? Remember how we liked CTRL+L because it immediately selected the URL, which was very handy for, like, then doing a CTRL+C to copy it and paste it somewhere else. He said: "Treating the text entered in the omnibar as a strict search query, even for terms that look like a URL, no quotes needed." He says: "As a .NET developer, libraries and frameworks with the something-or-other .NET naming convention turn up quite a bit in my searches." He says: "Chromium seems to have copied this, so that now Brave, Edge, and Chrome all behave like Firefox with CTRL+K forcing a search." And he said: "Thanks for everything you do. Patrick."

So Patrick, thank you. This was news to me, so I tried it, and it works perfectly. I previously shared my discovery, which I just mentioned, about CTRL+L. And I've now added CTRL+K to my bag of keyboard shortcuts because it leaves no confusion. You hit CTRL+K, and immediately the box - in my case I've just left my Firefox with the default Google search. And so it immediately comes up and like puts Google in front, confirming what engine it will send this to when I hit ENTER. So very cool tip, thank you.

I have no big SpinRite news, which at this stage is what we hope for. Everything continues to go well. I'm pushing forward on several fronts. I'm spending time in GRC's forums, watching as new people encounter SpinRite 6.1 for the first time. So I'm learning about their experiences which will be informing SpinRite's forthcoming FAQ.

One thing I've seen is that Linux users need SpinRite in a format that's directly usable to them. Requiring non-Windows users to briefly somehow use Windows, like someone else's Windows system, to have SpinRite create a bootable USB drive for them is something I always planned to work around, so not to require that. So I've been working on the tech to add direct bootable image downloading to GRC's servers so non-Windows users will be able to obtain a bootable drive image which they can copy directly to a USB drive, you know, just using "dd" in Linux, although I guess Ubuntu has a nice little drive imaging tool built in, which will allow them to boot their license copies of SpinRite.

I also have the beginnings of GRC's forthcoming email list system running that I know that a lot of our listeners are waiting for. Someone just said that he received a note from X, you know, Twitter X, that he would no longer, he'd be losing his DM'ing privileges if he didn't get more active or something. It's like, what? You know, Elon just insists on creating trouble for people. So I know that a lot of our listeners are only using Twitter so that they can send me notes from time to time, so I'll be fixing that. So I'm in the process of bringing up our email list system. And I plan to just maintain two lists, one specifically for weekly podcast announcements to our Security Now! listeners, and another for GRC-related news, you know, new software, updates, features, and so forth. So people will be able to join either or both as they choose. And I'm also working on SpinRite's documentation, which is coming along nicely.

Having learned from Microsoft last week that certificate reputation matters, even though they are deliberately mute about how exactly that reputation is earned, I presume it's a function of the exposure of a new certificate in the signing of non-malignant software. So right after last week's podcast I used this relatively new certificate that I've got to cosign GRC's top six most downloaded Windows freeware. In order of decreasing average daily download rates, the top six are ValiDrive, the DNS Benchmark, InSpectre, Securable, InControl, and Bootable.

What surprised me was that, for the first time ever, ValiDrive has taken the top slot to become GRC's most downloaded freeware with nearly 2,200 downloads per day. Anyway, taken together, those top six are being downloaded, in aggregate, 5,323 times per day. So now they're all carrying this new certificate whose reputation I want to earn, and I expect it'll become golden before long.

I should also mention something really weird that was just pointed out to me, and that is an update of Microsoft's Trusted Root Certificate Program. If I read this correctly, they're saying that the specialness of EV is being deprecated in August. They specifically say, in August of 2024, all certificates, EV or not, will be treated identically. So, what?

Anyway, I wrote to a good friend of mine at DigiCert, who back in the day were podcast listeners, I don't know if he still is, saying, am I reading this right? Is EV, you know, sort of in the same way that EV has been deprecated for SSL, right, like none of our browsers show us anything special about EV certs anymore. I'm wondering what's going on with EV code-signing. Maybe the same thing.

Anyway, I expect I'll have an answer from him, and I'll let everybody know next week. I'm certainly curious because, you know, I spent a month figuring out how to get dynamic, on-the-fly, server-side hardware security module cosigning for SpinRite, and now it looks like before it really gets used that much, maybe it's going to just be a nonstarter. Anyway, no, no.

Leo: We have EV certs for TWiT which we pay a lot of money for.

Steve: Now, those are TLS certs, though; right?

Leo: Yeah.

Steve: Yeah. And so I'm talking about code-signing EV.

Leo: Oh, for code-signing, of course, yeah, yeah, yeah.

Steve: Which Microsoft says, eh, we're not going to bother paying attention to that anymore.

Leo: So TLS EV certs continue.

Steve: Yes.

Leo: Oh, whew, okay. Scared me.

Steve: They continue, although - yeah. They continue, although they're like, you know, only if the user goes to look is it clear anymore that a site is an EV cert. It's, you know, I've stopped using them because there's no point. There's no benefit.

Leo: They don't show green anymore or anything like that.

Steve: Nope.

Leo: Yeah.

Steve: They don't show anything different. It's all disappeared from the chrome. So let's take our second break, Leo. And then we're going to do some feedback from our listeners before we get to a really interesting topic of what Apple has done with post-quantum crypto for their iMessage.

I am so glad you brought me back to the Humble Book Bundle because you know Cory Doctorow quite well.

Leo: Oh, good friend, yeah.

Steve: Yes. And this is, as far as I know, Cory's first incredibly affordable Humble Book Bundle. I mean, I'll just read the description of it from the page. I've got it in the show notes. It's humblebundle.com/books/cory-doctorow, and so it's easy to search and find.

The description reads: "Doctorow's visions of the future: Lose yourself in the visionary fiction of Cory Doctorow, the celebrated author and digital rights activist known for his masterful explorations of the intersection of tech and society. 'Little Brother' is a stark exploration of surveillance and authoritarianism in the backdrop of a major terrorist attack on San Francisco. 'Radicalized' features four distinct sci-fi novellas, each telling a gripping story inspired by today's technologies and societal trends. In 'Red Team Blues,' you'll follow along with a well-connected money laundering expert on the most dangerous and exciting gig he's ever taken on."

Leo: He's actually a forensic accountant is his hero in this series, and I love it. It's so good. He's so good. I love Cory.

Steve: Oh, very cool.

Leo: Yes.

Steve: It says: "Get all these and more, 18 books in total, and help support the Electronic Frontier Foundation with your purchase."

Leo: Oh, great.

Steve: So 18 books. As always, it's a pay what you feel it's worth, you know, as little as $18, a dollar each or more, in order to support Cory and the EFF. So anyway, I wanted to make sure everybody knew about this Doctorow Humble Book Bundle.

Leo: That's such a good thing. Thank you.

Steve: Yeah, very cool. So Tom Desmond said: "Hey, Steve. I'm just finishing listening to Episode 963" - so that was last week - "and I think I'm missing something on the cookie email link login loop. The" - he has it in quotes - "'absolutely secure' process seems to assume that no bad person has compromised the email. Think about this. Someone gains access to the email from wherever. They see an old email loop link. They go to the site and guess the username - maybe it's the same email address - and they get the link in the compromised email, and they are in. Am I missing something?"

And sadly, Tom, you're not. I hate it when I miss something that's so obvious. Tom is, of course, completely correct. The solution which I described last week of embedding the email link requestor's IP address and browser cookie into the link would absolutely and strongly prevent anyone who did not request the email-based login link from using it, if they were to just passively observe it in flight or at rest. That's nice. But as Tom points out, that doesn't solve the problem that we're still just as dependent upon email security as ever. Nothing prevents an attacker, as Tom says, who has the ability to monitor the user's email account, from themselves going to the site, requesting a login link from their IP location with their browser, then clicking the link that arrives in the compromised user's email. Whoops. And yes, they're logged in.

So this means that while there's no reason not to at least embed, for example, the requestor's browser cookie in the link, we don't really gain anything from doing that, and we remain dependent upon the security of email for all of our logon security, whether or

not it's passwordless using email only or passworded with the ubiquitous "I forgot my password" total security bypass. So thank you, Tom. And I should say Tom was just the first of a bunch of listeners who said, uh, Steve, I think you are solving the wrong problem. And they were right. So thank you, everybody.

Our longtime listener and early advertiser with the podcast, Alex Neihaus, he wrote: "Regarding email/password links in SN-962." So that's two weeks ago. He said: "If you send a link with a hash/IP address, et cetera, you eliminate the ability to copy the link and open it in another browser. It would also encourage clicking on links in email, something enterprises are trying hard to untrain users from doing." He said: "Also, if you did get such a link that you opened in Incognito mode, and the server expected a persistent session cookie, next time after restarting the browser you asked for the link in either normal mode or Incognito, it would not exist. Email links instead of passwords are a good idea," Alex thinks, "and can be made secure as you describe. But it does not fit with the way many people interact with browsers, especially those who use multiple browsers."

And of course Alex's point is a good one. Heavy use of email links for common logon would have the effect of "untraining" users against clicking on links in email. If the world were to switch over to that, and I don't think there's any danger of that happening, there would certainly be a lot more email link clicking going on. And it's also true that the tighter we make the anti-spoofing security, the greater the chance for rejecting a valid user. Which of course would upset them because this is the way they log on.

The presumption is that the user wishes to log on now, not later. So they would request a link, then immediately go to their email to find and execute the link, which would take them back to where they just were. But Alex is right that there are various things that could go wrong. So as we so often encounter, everything is a tradeoff. The great benefit of email link logon is that the user needs to remember and possess nothing, and they are able to log in from anywhere, as long as their email is secure, and they have access to it.

I think the best thing probably to come from the last few weeks of this discussion has been the recognition that passwords really only amount to logon accelerators, and that as long as every username and password logon opportunity is also accompanied by "the dog ate my password" bypass, nothing else we do to further increase logon security actually matters - not hardware dongles, not fingerprints, not one-time passwords, nothing. It all just amounts to security theater. This, in turn, implies that the security of our email is far more important than is commonly appreciated. And so I think that's another good thing to really come from this discussion.

And before we finish this, another listener sent this bit of fun. He wrote: "The Taco Bell app seems to have gone passwordless in favor of emailing you a link whenever to try to log in. It's not great user experience for me in that it's slow and prone to spam filtering. When you're trying to put in your order from the car on your way to the Bell, it's enough to make you drive to Chipotle instead."

So, yes. You want to make sure, if you're going to switch to - and I thought it was interesting that Taco Bell, the Taco Bell app now just wants to have you immediately confirm who you are by replying to a link in email. They don't want to create user friction. They apparently think that they are reducing it by not requiring the user to use a password all the time. But as this listener pointed out, they might lose some business.

Joel Clermont said: "There is a security-focused reason to enforce a max password length when hashing with Bcrypt." He said: "I recently discovered the recommendation and wrote up more details here." And so Joel provided a link to his write-up, which is located at "masteringlaravel.io," which was interesting. It turns out that Laravel uses Bcrypt as

its PBKDF password-hashing algorithm; and for reasons that defy understanding, the Bcrypt algorithm itself has a hard and fixed internal password length limit of 72 bytes.

Okay, now, normally we'd think that 72 bytes was way more than anyone might need. But unfortunately, many characters in non-Latin alphabets are encoded as three and even four bytes. So it would be possible for a non-Latin password to max-out Bcrypt's fixed 72-byte limit with just 18 characters. And again, that's probably enough.

But still, Joel makes a good point about there being possible exceptions to the boundless upper password length presumption. I'll just note that a simple solution to this problem for anyone who might be stuck using Bcrypt would be to first run the user's truly unlimited-length password through an SHA-256 hash, or even an SHA-512. Why not? And do it just once. SHA-512 will take a password of any length, even a long one using non-Latin characters, and reduce it to exactly 64 bytes, which can then be sent into Bcrypt to be strengthened against brute-force guessing. Which is why you use Bcrypt at all in the first place. Anyway, the cute little hack here is just stick a hash in front of something that can't take long enough passwords, and you completely solve that problem.

Leo: Does Argon2 have the same issue, or just Bcrypt?

Steve: No, no. It just was an early design decision in Bcrypt.

Leo: Bcrypt's kind of old, actually; isn't it? And kind of funky.

Steve: Yes. Yes. And one would not even start using it at this point.

Leo: Yeah.

Steve: Yeah. Earl Rodd said: "Regarding the discussion of Nevada's request for an injunction against Meta encrypting messages for minors in SN-963. As I listened, I was aware that the great missing piece is data. All of those quoted on both sides, privacy advocates and law enforcement advocates, have lots of opinions, but they present no data. Data like, you know, the number of times criminals were believed to go free because of encrypted messages stopping prosecution; or the number of times such crimes were not prosecuted at all due to encryption; or data like times that using unencrypted messages was known to lead to harm to minors. Is this really a case of balancing privacy, i.e., being pummeled with advertising, versus throttling the ability of law enforcement to find and prosecute serious crime?" He said: "I suspect it's not that simple. But without any data, who knows the actual tradeoff?"

Okay. I think Earl makes such a good point here. You know, we've been driven so far away from actual data collection that we appear to have forgotten that actual data could be collected and made available. You know, we're just never asking for it; right? We've, you know, we've become the "anecdotal example" culture, being slammed from one extreme to the other. And here's the problem: Expressing a strong opinion is easy. It excites. It's junk food for the mind. But actually collecting, tabulating, and evaluating data is difficult, time-consuming, and expensive work.

And the biggest problem is, as Earl suggests, the result of all that actual work probably doesn't support the extreme views of either side. So it's not in either side's best interest

to be presented with any of those pesky facts which would likely lead to some form of compromise policy. Better just to wave our arms around, attempting to terrify everyone, and jam through regulations that support an ideology rather than reality. It seems that too much of the world is working this way these days.

We've seen countless examples of how the move to an online digital world has very likely provided law enforcement agencies with a treasure trove of readily accessible, indexable, and searchable information the likes of which hasn't ever been known. It's never been known before. We've recently learned that these agencies are consumers of the information that commercial data brokers have been collecting about us for years. They're buying this data. Today, everyone leaves footprints and bread crumbs wherever they go online, and whatever they do.

If law enforcement knew what all of our various service providers know, and what our ISPs probably know about where we wander on the Internet - and we should assume that they do know any of that if they wanted to - then law enforcement knows pretty much everything about us, certainly more than was ever knowable about people before the Internet.

As a law-abiding citizen of the United States, which for the most part leaves its citizens alone unless some intervention is required, I'm all for law enforcement working behind the scenes to keep a lid on criminal behavior. That's what I want them to be doing. So given the unparalleled tools that are now available to aid in that work, it is difficult to get too worked up over the encryption debate. Law enforcement authorities can even know who we're talking to if they wish, even if it's much more difficult to peer into the content of those conversations. Although I also do not have any data to back up my opinion, my intuition is that we're already sitting with a workable compromise for both sides.

Could the privacy absolutists wish for more privacy? Absolutely. Are they going to get much more? Probably not. Do our law enforcement agencies wish they could also listen in on anyone's conversations? I'm sure they do. Are they going to be able to do that? Let's hope not because that would clearly be a step too far in the "Big Brother" direction. And again, they must already have access to far more data about us than they even know what to do with. It seems to me that the right balance has already been struck. But I think Earl's point is such a good one about hey, you know, like where's the data to back up any of this either way? So thank you for that.

Elliot Alderson said: "Hi, Steve. The issue with passwordless login the way you described is if someone only has email on their phone." He said: "I don't have email on my computers because I don't need it." Really. "If the IP and everything is baked into the link, I'm SOL. That also makes it very difficult to log in on other browsers." That's true.

So, okay. Having listened to everyone's thoughts about this, it's clear that the potential automation offered by an email link creates more problems than it solves. So the solution for email-only-based login is to return a visible, easy-to-transcribe, one-time six-digit token, like everyone who uses one-time passwords has become used to. Just email the token to the user and request that they enter the token into the browser they're currently using. If they receive the email on their phone, they have the advantage of seeing both the token and their browser page at the same time.

This eliminates the "untraining" about never clicking on a link, and it's not really that much more work for the user to type in six digits. And they still don't need to remember anything, their password or anything else. So I think, you know, stepping back from the link idea, you know, more automated though it was, it's got a lot of downsides which our listeners, astute as they are, have all pointed out. So thank you, everybody.

SecFan said: "Steve, in SN-963 yesterday, a listener had mentioned the idea of having a standardized way of documenting a site's password requirements so that they could be automated. In writing my own algorithmic password manager, named Passify" - that's a cute name - "that I messaged you about previously, I had the same thought and wanted Passify's algorithm to accommodate known requirements wherever possible. I discovered that Apple had put some thought into this and created 'Password Manager Resources' on GitHub." And that's what it's called, github.com/apple/password-manager-resources.

He said: "It includes a JSON format" - just sort of as I was suggesting last week - "for documenting password rules as well as a number of other helpful things for Password Managers to automate or semi-automate password management. The idea is not completely new. Apple's Safari browser has supported a 'passwordrules' HTML attribute that uses the same rule markup as the JSON to describe requirements for a while. Thanks again for all your work. Looking forward to the official release of the new SpinRite."

Anyway, it's cool that Apple has already done the work of creating some of this structure. And my comment last week was about it being a heavy lift to get the industry adoption, and that appears to be the case. This is now four years old, and it has remained somewhat obscure. Of course, if any website wanted to modernize, they could simply place a relatively high lower limit on password length and accept passwords of any greater length containing any characters. You know, just hash the thing a bunch, and just don't worry about how long it is or what the thing contains, and do it on the browser side, and you're good to go. And then on the server side, they could prevent endless brute forcing password guessing by putting up a roadblock if someone is misguessing the password to a site some number of times.

Rob said: "Security Now! question: After hearing your discussion of session cookies, I was reminded of my own security concern I've had for years. Seems I'm always logged into my Google account based on session cookies. And so it seems, if a hacker could grab those cookies from my computer, they'd now have access to all my emails and many other Google things. This," he says, "even with Google's Advanced Protection Program."

He says: "I believe I've even migrated to a new MacBook and was automatically logged in fine on the new MacBook," he says, "though my memory could be a bit off on that." And I don't think so. I think there's a lot of inter-browser synchronizing going on these days. He says: "Didn't seem there was any other verification that my hardware was the same, et cetera. Just doesn't feel very secure to me. Am I missing something?"

Okay. So first of all, a bit of clarification here because Rob used the term "session cookies," which I'm also seeing a lot of casual use about. When a cookie is set in a browser, the browser can be told how long to retain that cookie in the form of either an expiration date, at which point the cookie will then expire, or a maximum age. If either of those are provided, then the cookie is considered to be "persistent," and the browser will retain and return that cookie's value until it is refreshed, that is, the setting of the cookie is refreshed, the cookie is deleted by the user or by the website, which is able to do so, or that designated end-of-life is reached, and the cookie self-expires. Any cookies that have not expired will be retained from one browser session to the next. That is, they are persistent.

But I mentioned that the specification of these expirations was optional. If a cookie is set without any explicit expiration, then it is considered to be a session cookie, as in good for this browser session, because it will be retained only for the current browser session. It is explicitly retained only in RAM, and it is never written in any way to any permanent form of storage. Once the browser closes, the browser session ends, and the cookie will be forgotten.

So to Rob's question and uneasiness, it is the case that our browser's persistent cookies are now being retained over the long-term, and that's what keeps us logged into our various websites persistently from one boot-up of our computer to the next. And it's definitely the case that if bad guys could arrange to obtain those persistent login cookies, they could immediately impersonate us. In fact, this is exactly what that Firesheep Firefox add-on did many years ago. And this was one of the primary motivators for the move to always-present HTTPS.

Firesheep relied upon the mistake that most websites of the era were making. After authenticating with a username and password whose communication was protected by HTTPS, most popular sites would drop back to plain, lower overhead, unsecure, cleartext HTTP, figuring wrongly that the user's identity had been protected. They failed to take into account that the only way that user was remaining logged on from one HTTP page query to the next was that each browser query was accompanied by cookies; but that without the encryption protection provided by HTTPS, those cookies were being sent in the clear so that anyone who could eavesdrop could obtain them and immediately impersonate the user, obtaining parallel access to their currently logged-on web sessions.

So yes, Rob, our browser cookies do serve as persistent long-term authentication tokens; and anyone who might have some means for obtaining them could, indeed, impersonate their owner. Sometimes when logging into a site you'll see a "Trust this browser and remain logged in?" checkbox. This is useful when you do not want to leave a persistent cookie behind in that browser, for example, when logging into a shared Internet caf machine. You'll also want to carefully and explicitly log out of that machine once you're finished in order to at least invalidate that cookie, even if it isn't completely removed from the machine.

But that "Trust this browser and remain logged in?" question is offering to change the logged-on authentication cookie from a temporary session cookie to a long-term persistent cookie. And there is a difference. One sticks around. The other one should, if the browser's behaving itself correctly, and I know we talked about this in the past, there was a point at which the various important browsers, which at this point means Chromium-based browsers and Firefox and Safari, were being very careful never to write that transient true session-level cookie to permanent storage.

"Hi, Steve. Listened to SN-962 and wanted to provide a bit of additional information regarding your discussion on password requirements. I work for a company in the financial services sector, specifically insurance. I did want to note that many companies in this sector still rely on mainframes, and that is likely the case for many of these maximum password length limits and character restrictions due to IBM's fanatical backwards compatibility."

He said: "Several years ago, I interned for another company in this sector and was shocked at the requirements for passwords. Seven to eight characters only, and the only symbols allowed were the @ and the $ characters." He says: "How much of this is whose fault is a bit opaque to me even still, but I can say that it isn't entirely the mainframe's fault. I've seen that more 'modern'" - and he has that in quotes - "implementations allow up to a 40-character 'password phrase,' but still restrict a few characters. All that being said, you see these kinds of restrictions often in sectors that continue to rely on the mainframe, and other legacy platforms and products."

And I should also mention that a number of our other listeners have since sent me notes saying that, Steve, mainframes are not gone. They still exist. You know, they're not around the way they used to be, but they're not gone. So I appreciate the feedback that we still haven't moved very far. One of the most significant things we've seen and learned through the years of this podcast is the power and prevalence of inertia, which is everywhere, acting to keep things from changing.

One trick that I ought to mention that works quite well is to map a hashed value to some fixed alphabet. Say that a backend system can only accept eight characters of upper and lower case alpha, numbers, and as we saw in this case, an @ sign and a $ sign. Okay? So that's 26 lowercase and 26 uppercase. So now we're at 52. Ten digits brings us to 62. And two other characters brings the total alphabet size to 64.

Now, okay. First of all, the fact that the alphabet size turned out to be 64, you know, representable by exactly six binary bits, you know, 2^6 is 64, that should jump out at everyone. Okay. So let's take this case first. We want a web frontend to give its users unrestricted passwords. So it hashes whatever they provide, doesn't matter, absolutely doesn't matter, preferably using some contemporary PBKDF function like Argon2. The output will be 256 evenly distributed bits. We know that that's what hashes give us. So they are simply taken from either end, let's take them from the right end, you know, the least significant right end, if we regard this 256-bit value as a number.

So we'll take them six bits at a time. Each of those six bits is mapped back into one of those 64 characters. So after eight groups of six bits have been taken, those eight characters are then passed back to the creaky old mainframe as the user's password. The result is that the user is able to use whatever crazy long and special-character-filled password they may choose, and whatever they choose is first strengthened by a modern PBKDF to harden it against brute-force attack and then converted into a high-entropy eight-character password which meets the needs of the backend system.

Okay. But what if the backend system's password alphabet wasn't conveniently 64 characters, which allowed us to take, just simply take six bits at a time? Okay. So consider this: One way to visualize taking six bits at a time is dividing the large hash value by 64, which is to say the size of the alphabet. In a binary representation, division by two is a right shift of the bits. So dividing by 64 is shifting right by six bits. We can think of the bits that are shifted off to the right as the remainder of the division. So to extract a single character from an alphabet of 64, we are actually dividing the large binary hash value, we're doing a long division by 64 and taking the remainder, which will always range from 0 to 63, thus having one of 64 possible values.

So this means that we can extract characters of an alphabet of any size we wish from a large hash value by performing successive long division of the hash by the size of the alphabet we wish to extract where the remainder from each division will be the value that's mapped back into the alphabet. Anyway, I've always found this to be very cool and been fond of this solution since it provides high-entropy, evenly distributed characters from an alphabet of any size, extracted from large binary values, such as those that are conveniently produced from hashing.

Our second-to-the-last bit of feedback, we have a bit of a longer format piece, but this is significant and important, and there's some lessons here. And I've had several bits of conversation with him since, which I'll share. He said: "Hello, Steve. I've been a longtime listener, and even longer time SpinRite user, and I enjoy your weekly updates and common sense perspectives on all kinds of security topics. I want to share a rather long story with you and would like to hear your opinion about an interesting in-progress responsible vulnerability disclosure. Here goes." And it's actually not that long.

He said: "In October 2023 I came across a sign-in page of a high-profile company with 20-plus billion USD revenue, and more than 100 million monthly active users on their website. Not a small business. For my work, I sometimes just for fun regularly visit websites with the Developer Tools tab open. This time I noticed that the sign-in page of this particular website returned a Response Header that mentioned nginx/1.12.2 as the web server. In itself not a big problem, but it is better to hide this information from the public. However, if the page was really served by nginx 1.12.2, then the website would

have a big problem because this version of nginx is very old and has a number of critical and high-severity vulnerabilities that are known to be actively exploited.

"So being a good citizen, I wanted to tell them about the issues, just for the sake of improving their security and of course also for improving the security of their 100 million monthly active users." Apparently he was one of those. He said: "So I searched on the company's website for their policy for responsibly disclosing these vulnerabilities, but the best I could find was the email address for privacy-related questions. I wrote an email telling them about the vulnerabilities, included some screenshots for evidence, and asked some questions about their statements in their privacy policy and terms and conditions on protecting my information and just keeping the website bug-free. I was not looking for any financial compensation. I just wanted to make the world a better place. I soon received a response that a bug report had been filed, and that it could take up to 120 days for bug reports to be tested and validated. I received no answers for my questions about privacy.

"Fast-forward to February 2024, 118 days since their message that the bug report had been filed. I politely asked them about the status of the bug report, notified them that the vulnerabilities still existed, and asked them if my personal data is still safe with them. A week later I received an email from them saying that after review and risk rating by their information security team, the bug report qualifies for a bounty payment of $350. If I agree to this payout agreement, I also agree to keep my silence about the issue. My questions about privacy and trust and new questions about their responsible disclosure policies remain unanswered. In the meantime, the issues still exist, and a small tour around some of the other websites of the same company show similar problems.

"Now, the fact that these vulnerabilities are so easy to find, the fact that the reported web server is so old, the fact that it is so easy to remove a server banner, and the fact that it is still not fixed after more than four months makes me believe that this could be a honeypot." Well, he's being extremely generous. I don't believe that. I think that we're dealing with is a typical irresponsible company. Anyway, he says: "If it is, then I think that it is a very, very risky attempt of having a honeypot." You know, why would you have a honeypot as your main logon server? I doubt that's the case. He said: "Because it is easy to find, it is on pages where people enter their username and password, and if the media get hold of it, then the brand damage of such a high-profile company will be big."

He said: "I have asked them directly if this is a honeypot, but I did not get a reaction yet. What do you think about this? What should I do? Accept the payout and keep my silence? Should I also report the 30-plus other vulnerabilities on their websites and sign-in pages? Should I tell them that just by looking at the web server Response Headers, I know that they use a mix of nginx 1.12.2, Microsoft IIS 10.0, ASP.NET 4.0.30319, Shopify, WordPress, et cetera? Or should I just leave it up to them to find it all out? I mean, what could possibly go wrong? Kind regards; and again, I love your show. Keep going. Robert Blaakmeer."

Leo: Wow. He's a Good Samaritan.

Steve: So Robert, yeah. And I wrote, I said: "Robert, I agree that this is a conundrum. I took a quick peek at the version of nginx, and what immediately jumped out at me - as it would any researcher, nefarious or not - was CVE-2021-23017. It's a well-known, remotely exploitable, 'off by one' error which gives an attacker unauthenticated remote code execution capability."

Leo: Oy, oy, oy.

Steve: Uh-huh. And what's more, a three-year-old working Python proof-of-concept exploit is also publicly available.

Leo: Well, there you go.

Steve: I mean, it just doesn't get any worse.

Leo: Even a three year old could do it. Geez.

Steve: So Robert, I agree that you're right to be worried about them, and worried in general. It appears that any miscreant wishing to target this enterprise has everything they need to do so. Unfortunately, although you've done the right thing every step of the way, their ongoing negligence in dealing with this, which you had no way of anticipating upfront, has now compromised you, since you've acknowledged to them that you're in possession of information that could damage them - either reputationally or for use in an attack. And as we also know, attorneys can allege that, just by looking at their web server headers, you've obtained "non-public network hacking" information, and some judge who knows no better could be convinced.

Leo: It's happened.

Steve: It's happened before.

Leo: Yup.

Steve: I think I would have nothing further to do with them. The problem with a company that has become this large is that the people who you really should be talking to are unknown and inaccessible to you by design. Between you and them are layer upon layer of chair-warming functionaries who have no idea what you're even talking about. At this point, my recommendation would be for you to turn everything over to CISA and let them take it from there.

Leo: Perfect.

Steve: This completely discharges your responsibility in the matter while insulating you from any blowback since no one, not the company in question nor any judge, could fault you for confidentially and privately informing the U.S. Government's Cybersecurity & Infrastructure Security Agency of this potential critical vulnerability after having first given the company in question all of November, December, January, and February to deal with upgrading their servers.

CISA, the rare government agency that appears to actually have its act together, has a simple procedure for report filing. Go to www.cisa.gov/report. I also created a GRC shortcut so that even if someone forgets the agency's name, going to grc.sc/report will

always bounce you to that page at CISA. The page has several categories of problems to report. If you scroll down you'll find "Report software vulnerabilities or ICS vulnerabilities," which appears to fit best. If you go there, you get bounced over to CERT.org, which is the CERT Coordination Center at Carnegie Mellon University.

So I'd fill out that form to let the U.S. government powers that be know about this. One of the things you can do is attach a file to your submission. So I would send them your entire email thread with the company so they can see that you have acted responsibly at every step. You'll have done the right thing. And when CISA and CERT reach out to contact the company about their negligence over a four-year-old known critical vulnerability across all their web platforms, you can bet that they'll wind up speaking to someone who can effect the required changes. And at that point you'll have done everything you can, while insulating yourself as well as possible from any annoyance this company might feel over being made to help themselves.

Leo: And you're pretty sure, I mean, if you did this, the company might not be happy about it, but honestly I think you'd be safe from prosecution. Right?

Steve: Yes. I think you're as safe as you possibly could be. I should mention that he did that. He sent back to me afterwards that he took everything, he submitted it to them, he got a response saying "Thank you for the submission, and we'll take it from there."

Leo: Yeah. Excellent. That's probably the way to start in these things, I would say.

Steve: And finally - I really, I do, I think so, too. I mean, we've seen and have covered on the podcast, so our listeners have heard too many instances where the company goes after the responsible disclosure guy, saying, you know, you hacked us, you bad person. It's like, no. No no no no no no. Anyway, yeah. I would just...

Leo: Randal Schwartz did jail time for that; you know? I mean...

Steve: Yup.

Leo: It's terrible, yeah.

Steve: Yeah. It's wrong. Okay. And finally, for next week, we have the first appearance of a zero-click GenAI application worm. As I mentioned at the top of the show, Ben Nassi sent me a tweet. He said: "Hi, Steve. I'm Ben Nassi, a former researcher at the Ben-Gurion University and Cornell Tech, and the author of 'Video-based Cryptanalysis and Lamphone' which you previously covered in your podcasts. I just published a new research paper on a worm that targets GenAI applications. We named it Morris II. Guess why?" He said: "I think the audience of Security Now! will love to hear more about it. And keep on with the great podcast that you and Leo are doing beyond 999. By the way, the research was also published on Bruce Schneier's blog."

Leo: Oh, that's pretty cool, yeah.

Steve: So I suspect that next week's podcast will be titled "Morris II." So stay tuned for a look at the "worminization" of GenAI, where I'm sure we'll be answering the question, "What could possibly go wrong?"

Leo: Now, PQ Trois.

Steve: So we can guess, we know, that PQ stands for Post-Quantum. And of course we'd be right. So is this Apple's third attempt at a post-quantum protocol, after one and two somehow failed or fell short?

Leo: What? No.

Steve: You know, PQ3, what happened to PQ1 and PQ2? No, Apple has apparently invented levels of security.

Leo: Oh.

Steve: And put themselves at the top of the heap. So PQ3 offers what they call "Level 3 Security."

Leo: Oh.

Steve: So here's how SEAR, S-E-A-R, Apple's Security Engineering and Architecture group, introduced this new protocol in their recent blog posting. They write: "Today we are announcing the most significant cryptographic security upgrade in iMessage history with the introduction of PQ3, a groundbreaking post-quantum cryptographic protocol that advances the state of the art of end-to-end secure messaging. With compromise-resilient encryption and extensive defenses against even highly sophisticated quantum attacks, PQ3 is the first messaging protocol to reach what we call Level 3 Security, providing protocol protections that surpass those in all other widely deployed messaging apps. To our knowledge, PQ3 has the strongest security properties of any at-scale messaging protocol in the world."

Well. So they're proud of their work, and they've decided to say that not only will iMessage be using an explicitly quantum-safe encrypted messaging technology, but that theirs is bigger than - I mean better than - anyone else's. Since so far this appears to be as much a marketing promotion as a technical disclosure, it's worth noting that they have outlined the four levels of messaging security which places them alone at the top of the heap.

Apple has defined four levels of messaging with levels 0 and 1 being pre-quantum - offering no protection from quantum computing breakthroughs being able to break their encryption - and levels 2 and 3 being post-quantum. Level 0 is defined as "No end-to-end encryption by default," and they place QQ, Skype, Telegram, and WeChat into this Level 0 category. Now, this causes me to be immediately skeptical, you know, of this as being marketing nonsense, since although I have never had any respect for Telegram's ad hoc basement cryptography, which they defend not with any sound theory, but by offering a reward to anyone who can break it, we all know that Telegram is encrypted, and that everyone using it is using it because it's encrypted. So lumping it into Level 0 and saying

that it's not encrypted by default is, at best, very disingenuous; and I think it should be beneath Apple. But okay.

Level 1 are those pre-encryption algorithms, I'm sorry, pre-quantum because 0 and 1 are both pre-quantum, you know, non-quantum safe, so Level 1 are those pre-quantum algorithms that Apple says are encrypted by default. The messaging apps they have placed in Level 1 are Line; Viber; WhatsApp; and the previous Signal, you know, before they added post-quantum encryption, which we covered a few months ago; and the previous iMessage, before it added PQ3, which actually it hasn't quite added yet. But it's going to.

Now we move into the quantum-safe realm for levels 2 and 3. Since Signal beat Apple to the quantum-safe punch, it's sitting all by its lonesome at Level 2 with the sole feature of offering post-quantum crypto, with end-to-end encryption by default. Presumably, as Signal's relatively new PQXDH protocol moves into WhatsApp and other messaging platforms based on Signal's open technologies, those other messaging apps will also inherit Level 2 status, lifting them from the lowly levels of 0 and 1. But Apple's new PQ3 iMessaging system adds an additional feature that Signal lacks, which is how Apple granted themselves sole dominion over Level 3. Apple's PQ3 adds ongoing post-quantum rekeying to its messaging, which they make a point of noting Signal currently lacks.

This blog posting was clearly written by the marketing people, so it's freighted with far more self-aggrandizing text than would ever be found in a technical cryptographic protocol disclosure, which this is not. But I need to share some of it to set the stage, since what Apple feels is important about PQ3 is its attribute of ongoing rekeying.

So to that end, Apple reminds us: "When iMessage launched in 2011, it was the first widely available messaging app to provide end-to-end encryption by default, and we've significantly upgraded its cryptography over the years. We most recently strengthened the iMessage cryptographic protocol in 2019 by switching from RSA to Elliptic Curve cryptography, and by protecting encryption keys on device with the Secure Enclave, making them significantly harder to extract from a device even for the most sophisticated adversaries. That protocol update went even further, with an additional layer of defense, a periodic rekey mechanism to provide cryptographic self-healing, even in the extremely unlikely case that a key ever became compromised."

Again, extremely unlikely case that a key ever became compromised, they acknowledge. But now they have rekeying, so it's possible. "Each of these advances were formally verified by symbolic evaluation, a best practice that provides strong assurances of the security of cryptographic protocols." And about all that I agree.

Okay. So Apple has had on-the-fly ongoing rekeying in iMessage for some time, and it's clear that they're going to be selling this as a differentiator for PQ3 to distance themselves from the competition. After telling us a whole bunch more about how wonderful they are, they get down to explaining their level system and why they believe PQ3 is an important differentiator. Here's what they say, and I skipped all the other stuff.

They said: "To reason through how various messaging applications mitigate attacks, it's helpful to place them along a spectrum of security properties. There's no standard comparison to employ for this purpose, so we lay out our own simple, coarse-grained progression of messaging security levels. We start with classical cryptography and progress toward quantum security, which addresses current and future threats from quantum computers. Most existing messaging apps fall either into Level 0, no end-to-end encryption by default and no quantum security; or Level 1, with end-to-end encryption by default, but no quantum security.

"A few months ago, Signal added support for the PQXDH protocol, becoming the first large-scale messaging app to introduce post-quantum security in the initial key establishment. This is a welcome and critical step that, by our scale, elevated Signal from Level 1 to Level 2 security.

"At Level 2, the application of post-quantum cryptography is limited to the initial key establishment, providing quantum security only if the conversation key material is never compromised. But today's sophisticated adversaries already have incentives to compromise encryption keys because doing so gives them the ability to decrypt messages protected by those keys for as long as the keys don't change. To best protect end-to-end encrypted messaging, the post-quantum keys need to change on an ongoing basis to place an upper bound on how much of a conversation can be exposed by any single point-in-time key compromise, both now and with future quantum computers.

"Therefore, we believe messaging protocols should go even further and attain Level 3 security, where post-quantum cryptography is used to secure both the initial key establishment and the ongoing message exchange, with the ability to rapidly and automatically restore the cryptographic security of a conversation, even if a given key becomes compromised."

Okay. It would be very interesting to hear Signal's rebuttal to this, since it's entirely possible that this is mostly irrelevant, largely irrelevant marketing speak. It's not that it's not true, and that continuous key rotation is not useful. We've talked about this in the past. Key rotation gives a cryptographic protocol a highly desirable property known as Perfect Forward Secrecy. Essentially, the keys the parties are using to protect their conversation from prying eyes are ephemeral. A conversation flow is broken up and compartmentalized by the key that's in use at the moment. But the protocol never allows a single key to be used for long. The key is periodically changed. The reason I'd like to hear a rebuttal from Signal is that their protocol, Signal's protocol, has always featured perfect forward secrecy. Remember "Axolotl"?

Leo: Yeah.

Steve: Yes.

Leo: Which is for the same purpose as rekeying.

Steve: Yes. It is rekeying.

Leo: Oh.

Steve: Here's what Wikipedia says. I'm quoting from Wikipedia. "In cryptography, the Double Ratchet Algorithm, previously referred to as the Axolotl Ratchet, is a key management algorithm that was developed by Trevor Perrin and Moxie Marlinspike in 2013."

Leo: Oh.

Steve: Uh-huh. "It can be used as part of a cryptographic protocol to provide end-to-end encryption for instant messaging. After an initial key exchange, it manages the ongoing renewal and maintenance of short-lived session keys. It combines a cryptographic so-called 'ratchet' based on the Diffie-Hellman key exchange and a ratchet based on a key derivation function such as a hash function, and is therefore called a 'double ratchet.' The algorithm provides forward secrecy for messages, and implicit renegotiation of forward keys, properties for which the protocol is named."

Right. In 2013. In other words, it would be nice to hear from Signal, since Apple appears to be suggesting that they alone are offering the property of perfect forward secrecy for quantum-safe messaging when it certainly appears that Signal got there 11 years ago. This is not to say that having this feature in iMessage is not a good thing. But it appears that Apple may not actually be alone at PQ's Level 3, much as they would like to be. So when do we see it from Apple?

They write: "Support for PQ3 will start to roll out with the public release of iOS 17.4, iPadOS 17.4, macOS 14.4, and watchOS 10.4, and is already in the corresponding developer preview and beta releases. iMessage conversations between devices that support PQ3 are automatically ramping up to the post-quantum encryption protocol. As we gain operational experience with PQ3 at the massive global scale of iMessage, it will fully replace the existing protocol within all supported conversations this year."

Okay. So now I want to share Apple's description of the design of PQ3. It includes a bunch of interesting details and also something that Telegram has never had, which is multiple formal proofs of correctness. Since we're now able to do this, it's a crucial step for trusting any newly created cryptographic system. Here's what Apple wrote.

They said: "More than simply replacing an existing cryptographic algorithm with a new one, we rebuilt the iMessage cryptographic protocol from the ground up to advance the state of the art in end-to-end encryption, and to deliver on the following requirements." Five of them. "Introduce post-quantum cryptography from the start of a conversation, so that all communication is protected from current and future adversaries. Two, mitigate the impact of key compromises by limiting how many past and future messages can be decrypted with a single compromised key. Three, use a hybrid design to combine new post-quantum algorithms with current Elliptic Curve algorithms" - which Signal has already done - "ensuring that PQ3 can never be less safe than the existing classical protocol. Four, amortize message size to avoid excessive additional overhead from the added security." And as we'll see, there really is some. "And five, use formal verification methods to provide strong security assurances for the new protocol."

Okay. They say: "PQ3 introduces a new post-quantum encryption key in the set of public keys each device generates locally and transmits to Apple servers as part of iMessage registration." So devices generate a post-quantum key on device, store it in the Secure Enclave, and then send the public part of that to Apple. "For this application, we chose to use Kyber post-quantum public keys, an algorithm that received close scrutiny from the global cryptographic community and was selected by NIST as the Module Lattice-based Key Encapsulation Mechanism standard, or ML-KEM. This establishes sender devices to obtain a receiver's public keys and generate post-quantum encryption keys for the very first message, even if the receiver is offline." Because, you know, Apple has the matching public key, which is all they need to perform the handshake. They said: "We refer to this as initial key establishment."

They said: "We then include, within conversations, a periodic post-quantum rekeying mechanism that has the ability to self-heal from key compromise" - which that's their fancy term for just moving forward - "and protect future messages. In PQ3, the new keys sent along with the conversation are used to create fresh message encryption keys that cannot be computed from past ones, thereby bringing the conversation back to a secure

state, even if previous keys were extracted or compromised by an adversary. PQ3 is the first large-scale cryptographic messaging protocol to introduce this novel post-quantum rekeying property."

Leo: [Buzzer sound]

Steve: And I don't - yes.

Leo: [Buzzer sound]

Steve: Exactly. This is where we need to hear from Signal because, eh, maybe not. Maybe not ever since Signal added post-quantum to their system. They say: "PQ3 employs a hybrid design that combines Elliptic Curve crypto with post-quantum encryption, both during the initial key establishment and during rekeying. Thus, the new cryptography is purely additive" - which is what Signal did - "and defeating PQ3 security requires defeating both the existing classical ECC crypto and the new post-quantum primitives. It also means the protocol benefits from all the experience we accumulated from deploying the ECC protocol and its implementations," even though they also said they redesigned it from scratch, so maybe it actually doesn't apply. But it makes nice marketing speak.

"Rekeying in PQ3 involves" - oh, yeah. No, it's the next piece I want to get to. But they said: "Rekeying in PQ3 involves transmitting fresh public key material in-band with the encrypted messages that devices are exchanging. The new public key based on Elliptic Curve Diffie-Hellman (ECDH) is transmitted inline with every response. The post-quantum key used by PQ3 has a significantly larger size" - oh, and does it ever, 2K, we'll talk about that in a minute - "than the existing protocol, so to meet our message size requirements we designed the quantum-secure rekeying to happen periodically rather than with every message. To determine whether a new post-quantum key is transmitted, PQ3 uses a rekeying condition that aims to balance the average size of messages on the wire, preserve the user experience in limited connectivity scenarios, and keep the global volume of messages within the capacity of our server infrastructure." Which is all a fancy way of saying we send it a little bit at a time.

Anyway, I'll just interrupt here also to note that it seems likely that PQ3 is rotating its quantum keys more frequently than Signal. I don't recall the details of Signal's ratchet, and it may have changed since we last looked at it. But it might also be that this is a distinction without a difference. In the case of Signal's ratchet, it was designed, not only to provide useful forward secrecy, but as a means for resynchronizing offline asynchronous end points.

The reason I suggest that it may be a distinction without a difference is that these key compromises are purely what-ifs. No one knows of any scenario where that could actually happen. The only reason Apple is mentioning it is that they have a way of sidestepping this as a problem, so now it's a problem. Okay. You know, if anyone did have a way of sidestepping it, then this problem, you know, the problem would be eliminated. You know, it's a bit like saying "My password is way stronger than yours because it's 200 characters long." Okay, that's good, but does it really matter?

Anyway, Apple continues: "With PQ3," they say, "iMessage continues to rely on classical cryptographic algorithms to authenticate the sender and verify the Contact Key Verification account key because these mechanisms cannot be attacked retroactively with

future quantum computers." In other words, they didn't bother upgrading that to post-quantum because they're not quantum vulnerable, in the same way that hashes are not.

"To attempt to insert themselves in the middle of an iMessage conversation, an adversary would require a quantum computer capable of breaking one of the authentication keys before or at the time the communication takes place. In other words, these attacks cannot be performed in a Harvest Now, Decrypt Later scenario. They require the existence of a quantum computer capable of performing the attacks contemporaneously with the communication being attacked. We believe any such capability is still many years away. But as the threat of quantum computers evolves, we will continue to assess the need for post-quantum authentication to thwart such attacks.

"Our final requirement for iMessage PQ3 is formal verification, a mathematical proof of the intended security properties of the protocol. PQ3 received extensive review from Apple's own multi-disciplinary teams in Security Engineering and Architecture, as well as from some of the world's foremost experts in cryptography. This includes a team led by Professor David Basin, head of the Information Security Group at ETH Zurich and one of the inventors of Tamarin, a leading security protocol verification tool that was also used to evaluate PQ3; as well as Professor Douglas Stebila from the University of Waterloo, who has performed extensive research on post-quantum security for Internet protocols.

"Each took a different but complementary approach, using different mathematical models to demonstrate that as long as the underlying cryptographic algorithms" - and they actually meant, yeah, okay, right, algorithms, like the post-quantum NIST-approved algorithms themselves. "As long as the underlying algorithms remain secure, so does PQ3." In other words, the protocols built on top of those algorithms will also be secure. They said: "Finally, a leading third-party security consultancy supplemented our internal implementation review with an independent assessment of the PQ3 source code, which found no security issues.

"In the first mathematical security analysis of the iMessage PQ3 protocol, Professor Douglas Stebila focused on so-called game-based proofs. This technique, also known as reduction, defines a series of 'games' or logical statements to show that the protocol is at least as strong as the algorithms that underpin it. Stebila's analysis shows that PQ3 provides confidentiality even in the presence of some compromises against both classical and quantum adversaries, in both the initial key establishment and the ongoing rekeying phase of the protocol.

"The analysis decomposes the many layers of key derivations down to the message keys and proves that, for an attacker, they are indistinguishable from random noise. Through an extensive demonstration that considers different attack paths for classical and quantum attackers in the proofs, Stebila shows that the keys used for PQ3 are secure as long as either the Elliptic Curve Diffie-Hellman problem remains hard or the Kyber post-quantum KEM remains secure."

And Apple then inserts a quote from Professor Douglas Stebila, which reads: "The iMessage PQ3 protocol is a well-designed cryptographic protocol for secure messaging that uses state-of-the-art techniques for end-to-end encrypted communication. In my analysis using the reductionist security methodology, I confirmed that the PQ3 protocol provides post-quantum confidentiality, which can give users confidence in the privacy of their communication even in the face of potential improvements in quantum computing technology. Signed, Professor Douglas Stebila."

Leo: It would have been much better in a German accent. I'm just saying. You really want people to believe a scientist, do it in a German accent.

Steve: That's true. That's true. And then Apple continues: "In the second evaluation, titled 'A Formal Analysis of the iMessage PQ3 Messaging Protocol,' Professor David Basin, Felix Linker, and Dr. Ralf Sasse at ETH Zurich..."

Leo: Ah, now we are talking here. We've got some real German scientists. Swiss is even better.

Steve: Okay. Well, in that case you're going to read the quote when we get to it in a minute. "...[U]se a method called 'symbolic evaluation.' As highlighted in the paper's abstract," writes Apple, "this analysis includes a detailed formal model of the iMessage PQ3 protocol, a precise specification of its fine-grained security properties, and machine-checked proofs using the state-of-the-art symbolic Tamarin prover."

Leo: Oh-ho.

Steve: Yeah, you've got to have that. "The evaluation yielded a fine-grained analysis of the secrecy properties of PQ3, proving that 'In the absence of the sender or recipient being compromised, all keys and messages transmitted are secret,' and that 'Compromises can be tolerated in a well-defined sense where the effect of the compromise on the secrecy of data is limited in time and effect,' which confirms," writes Apple, "that PQ3 meets our goals." And they quote Professor Basin. Leo, take it away.

Leo: "We provide a mathematical model of PQ3 as well as prove its secrecy and authenticity properties using a verification tool for machine-checked security proofs. We prove the properties even when the protocol operates in the presence of very strong adversaries who can corrupt parties or possess quantum computers and therefore defeat classical cryptography. PQ3 goes beyond Signal with regards to post-quantum defenses. In PQ3, a post-quantum secure algorithm is part of the ratcheting and used repeatedly, over and over, rather than only once in the initialization as in Signal. Our verification provides a very high degree of assurance that the protocol as designed functions securely, even in the post-quantum world." Doesn't that sound more credible?

Steve: Oh, Professor.

Leo: Don't you believe that?

Steve: Professor. Now, what I found disturbing about the quote, not Leo's rendition...

Leo: Well, that's disturbing in its own right.

Steve: They had the professor say "PQ3 goes beyond Signal with regards to post-quantum defenses."

Leo: Yeah, that's really interesting.

Steve: The dig at Signal was entirely unnecessary and gratuitous. And I don't understand why Apple apparently feels so threatened by Signal. Oh, wait, yes, I do. Signal is open, open design, open source, and entirely cross-platform. Anyone's conversations can be protected by Signal on any platform they choose, whereas iMessage, just like everything else Apple does, is all about platform lock-in.

So is PQ3 any reason to choose iMessage over Signal? No. When we're talking about cryptography, we've learned that there's nothing wrong with adding a belt to those suspenders. After all, that's why Apple copied Signal in adding post-quantum crypto to existing and well-proven pre-quantum crypto. Belt and suspenders. And so, if your work model allows you to be stuck within Apple's closed ecosystem, then you can be confident that iMessage will be secure against any current and future surprises. I have no doubt that Apple did all of this right. But you'll certainly be secure enough using Signal, and you'll have the benefit of also having far more freedom.

Leo: And the ability to talk to the rest of the world.

Steve: Exactly.

Leo: Yeah.

Steve: In a secure fashion. Finally, the paper drops into lots of interesting detail that I won't drag everyone through since we already have the essence of what's going on. But I did want to share one piece of the techie bits since I think it's interesting regarding the overhead that Apple has introduced by their more or less continuous rekeying. And actually we find out how continuous. So keep in mind that text messages are often no longer than old school SMS tweets, or shorter. You know, sometimes just a few words; right?

Apple explains: "To limit the size overhead incurred by frequent rekeying while preserving a high level of security, the post-quantum KEM is instantiated with Kyber768. Unlike the IDS-registered public keys used for the initial key establishment, ratcheting public keys are used only once to encapsulate a shared secret to the receiver, significantly limiting the impact of the compromise of a single key. However, while a 32-byte Elliptic Curve Diffie-Hellman-based ratchet overhead is acceptable on every message, the post-quantum KEM ratchet increases the message size by more than two kilobytes."

Leo: Ooh.

Steve: Yeah.

Leo: 2,000 characters in English, in ASCII.

Steve: Yes.

Leo: That's lots.

Steve: In an "Okay, thanks Mom, I'll be there soon" message.

Leo: Yeah.

Steve: "To avoid visible delays in message delivery when device connectivity is limited, this ratchet needs to be amortized over multiple messages." In other words, stretched out.

Leo: Spread out, yeah.

Steve: Yeah. "We therefore implemented an adaptive post-quantum rekeying criterion that takes into account the number of outgoing messages, the time elapsed since last rekeying, the current connectivity conditions. At launch, this means the post-quantum ratchet is performed approximately every 50, five zero, messages; but the criterion is bounded such that rekeying is always guaranteed to occur at least once every seven days, so at least weekly. And as we mentioned earlier, as the threat of quantum computers and infrastructure capacity evolves over time, future software updates can increase the rekeying frequency while preserving full backward compatibility."

Okay. So now everyone knows as much about PQ3 as is necessary. Apple has followed Signal by adding a believed-to-be-strong post-quantum crypto algorithm to their built-in iMessaging platform, which will be arriving with iOS's next major update. And if you're talking to somebody who also has been able to upgrade their device to 17.4, then your conversation will be post-quantum encrypted. Apple has also taken the welcome step of having their largely new iMessaging protocol formally proven by highly qualified academic algorithm researchers with wonderful accents. It's only a bit sad that Apple clearly feels so threatened by Signal.

Leo: Celebrity accents impersonated. Actually, the most important thing is this is Kyber. All three I think of the NIST protocols are Kyber, as well; right? There was one that failed.

Steve: Yeah, one was like in consideration, and...

Leo: It crashed.

Steve: Problems were found, yeah.

Leo: Have these been approved yet? Or is NIST still testing?

Steve: We're still - we're in the late phases of this.

Leo: Okay.

Steve: So, I mean, again, everyone is being really careful because this is where you want to be careful. This is where, you know, care matters.

Leo: So they're using Kyber768 with perfect forward secrecy. Those are the two things that probably, you know, should matter to you. Whether they're better than Signal is a pretty big and loaded question, obviously. It's a marketing question, I think.

Steve: Yeah. And I'll be surprised after this if we don't hear something from Signal, you know, in their own well-crafted accent.

Leo: Yeah. They don't, I mean, honestly, they don't need to prove anything.

Steve: No.

Leo: But it would probably be appropriate for them to say, hey, you know, just so you know, we're not PQ2. We provide exactly as good protection as Apple does. And you shouldn't, you know, consider us anything less, despite what Apple may say.

Steve: Right.

Leo: Now, all of this is presuming that at some point we'll get quantum computing, which is still a long shot. I mean, it's not - we're not close by any means.

Steve: Yes. The only - at this point what everyone is doing is considering the harvest now, decrypt later scenario.

Leo: Right.

Steve: They're wanting to move us into crypto, I mean, it's like, why not? We've got the computing power. We've got the bandwidth. You know, obviously the keys are much longer in order to be as secure as we need them to be.

Leo: Right.

Steve: So there is some cost. But it's cost we can now afford. So let's move the entire world now so that, you know, the NSA server farm in Utah can have its next expansion and continue storing stuff, but probably unable to decrypt it once quantum computing does happen.

Leo: You know, you might wonder, well, when is - so we're talking, yeah, in 20, 30 years, if they still have your messages, and if you still care, absent any kind of post-quantum crypto, they might be able to crack it. But that's pretty much a long way off. It's not just around the corner.

Steve: Yeah, I mean, we have to imagine that what they're storing are not people saying they'll be home soon for dinner.

Leo: Right.

Steve: You know, that they're storing...

Leo: Well, they are storing that. They're storing everything. That's the thing.

Steve: They're storing that, too. But also, you know, China's conversations with their advanced persistent threat groups and where the money is being wired in order to fund them and all that. I mean, all of that that is not yet post-quantum encrypted is, you know, [crosstalk].

Leo: It's hard to imagine anything, though, that would be decrypted in a few decades that would have anything other than historical interest; right?

Steve: I agree.

Leo: Anyway, it's, you know what, you can do it, we've got the means, why not go ahead and do it.

Steve: Yeah, it's like Microsoft publishing the source for MS-DOS finally. It's like, okay.

Leo: Thanks.

Steve: Does anyone care? Yeah, thanks a lot.

Leo: You might care. But other than that, yeah. No, and I appreciate you kind of addressing the marketing kind of fluff in here from the technological point of view because I think it is. It doesn't - it's not needed. It wasn't necessary, Apple.

Steve: I was disappointed. It's like, wait a minute; you know? They have a ratchet. And Signal hasn't told us how often they deploy it.

Leo: Right.

Steve: But, you know, again, nobody doesn't think that Signal is secure. Everybody knows that Signal, I mean, it's the standard.

Leo: It's the gold standard.

Steve: Yes.

Leo: I think that's why Apple's attacking them, frankly. They're the standard. They're the gold standard. And there's nothing in here that makes them any less of a gold standard for good post-quantum crypto.

Steve: Because Apple went to all this work - it's funny, Leo, I was also thinking about the goggles, whatever the hell they are, you know, where - and I guess I listened to you on Sunday. I got this - I came away with a clear sense that Apple had gone too far.

Leo: Yeah.

Steve: That they really, they far over-engineered those things. And I'm beginning to wonder maybe if that's what Apple has become is like a way over-engineering company, because you could argue that there is some cost to mistakenly or needlessly over-engineer things. And the very fact that they're having to amortize their own post-quantum rekeying over at least 50 messages in order not to bog it down too much suggests that, you know, that this was purely a marketing point.

Leo: Wow. Good point, yeah. But it may be more than just marketing. It may also be a message to governments, especially China and Russia...

Steve: Don't bother.

Leo: ...but also the U.S. Yeah. No, no, we're committed to end-to-end encryption, so knock it off. I think there's some of that there, as well. And for that I applaud them. I think they're doing...

Steve: That we're not softening anything.

Leo: We're going, quite the opposite, we're going the other direction.

Steve: Yeah.

Leo: Yeah. All right. Hey, Steve, thank you very much. I have one more thing I would say is I think Telegram is encrypted, but you have to choose it. I don't think all Telegram messages are encrypted, at least that's how it used to be.

Steve: Yes. And I agree with you. I don't think that Apple is lying about saying that it's not encrypted by default.

Leo: By default is the thing, yeah.

Steve: But who doesn't turn that on? And so it's like, you know...

Leo: You have to choose an encrypted message, yeah.

Steve: Yeah. And in fact I would argue that having Level 0 is dumb because you either have end-to-end encryption or you don't.

Leo: Or you don't; right.

Steve: So, you know, why have zero and put some people in that category? Oh, you have to turn it on. Okay, so everyone does. It's the first thing they do...

Leo: Right.

Steve: ...when they load it is they turn on the encryption.

Leo: Yeah, yeah.

Steve: Now I'm sure [indiscernible] up, how would you like to be encrypted? Uh, okay.

Leo: Okay. Oh, yeah.

Steve: Good idea. I think I do.

Leo: And then what's for dinner?

Steve: Why do you think I got this dumb thing?

Leo: Steve Gibson, GRC.com. That's the place to go.