The Endless Journey to IPv6

Description: Apple proposes 45-day maximum certificate life. Please, no. SEC fines four companies for downplaying their SolarWinds attack severity. Google adds five new features to Messenger, including inappropriate content. Does AI-driven local device-side filtering resolve the encryption dilemma forever? The very nice-looking "Session" messenger leaves Australia for Switzerland. Another quick look at the question of the EU's software liability moves. Fake North Korean employees were found to install backdoor malware. How to speed up an SSD without using SpinRite. Using ChatGPT to review and suggest improvements in code. And Internet governance has been trying to move the Internet to IPv6 for the past 25 years, but the Internet just doesn't want to go. Why not? And will it ever?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-998.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-998-lq.mp3

SHOW TEASE: This week on Security Now!, Apple wants to shorten the life of your SSL certificate. Steve's up in arms about that. We'll talk about a very nice new messenger program that Steve says may even be better than Signal. And then we'll take a look at IPv6. Whatever happened to it? It looks like it's going to be another 20 years. Steve explains why that's okay. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 998, recorded Tuesday, October 29th, 2024: The Endless Journey to IPv6.

It's time for Security Now!, the show where we cover the latest security and privacy news, keep you up to date on the latest hacks, and we get a little sci-fi and health in there, as well, because Mr. Steve Gibson is what we call a polymath. He is fascinated by it all. Hi, Steve.

Steve Gibson: Hello, my friend. Speaking of science fiction, it's not in the show notes, but I am at 40% into...

Leo: "Exodus."

Steve: ...Hamilton's "Exodus." And I have to say I'm glad it's long because it's come together. There are so many things. I mean, I could talk to John Slanina, I guess, because I know he's on his...

Leo: He's probably finished it by now.

Steve: No, he's in his re-read already.

Leo: Oh, wow.

Steve: He shot through it, and he's reading it again. He said the second time through he knows who the people are so - because, I mean, it's, you know, Hamilton doesn't write thin sci-fi. But there are so many really interesting concepts like - and this isn't - there's no spoilers here. Faster-than-light travel is never invented, so we never have FTL. But what we have is, well, okay. Also this is set, like, 50,000 years in the future. So we're in an environment where there are the so-called "remnant wars" that have left behind, like, dead planets and derelict, like, hugely up-armored technologies. And we've gone so far in the future that we've lost some of the knowledge that was, you know, during the peak war time. So they're, like, finding this stuff that they don't understand, but they kind of like, you know, give it power and see what it does.

And but the other thing is cool is that there are - this one elevated old, old race created what's known as the "gates of heaven." And they're gates which draw on a huge source of energy to bring their ships up to 0.9999 light speed. I mean, like, right up to c, but not quite because you can't actually, you know, it takes infinite energy to get all the way there. But what it does is it of course creates huge relativistic time compression so that the people who are traveling at .99999 of c, they, you know, it was a one-week trip for them. Meanwhile, four generations are gone.

So anyway, but again, it's just it's this whole 'nother - he did it again. There's a whole 'nother rich tapestry of really good hard sci-fi, Hamilton style. And as I said, I'm at 40%. I have no idea, can't even begin to guess what's going to happen. I have no idea what's going to happen. But I do not want it to end because it's just really solid entertainment.

Leo: It's like a locomotive, I think, where it starts slow, maybe the first couple of pages the wheels spin a little bit. And then it chug, chug, chug, chug, chug, chug, chug, like that.

Steve: First couple - first third of the book. Yeah, it was - you have to just sort of hold your breath. And, you know, you do want to read the early history because he, like, summarizes the history in a preface.

Leo: I listened to that, and then it got to the dramatis personae. And, now, I'm listening.

Steve: Yes.

Leo: It went on for a while.

Steve: Oh, my god.

Leo: And all the people...

Steve: The problem is you don't have any - there's no reference point.

Leo: Right.

Steve: Everybody is defined in their relationships to each other. It was like, okay, why...

Leo: It's people's names. So that's the part where the wheels are spinning, but the train is not moving forward much. But I just got past that. So I'm looking forward to the journey.

Steve: Oh, let me tell you, I mean, since I don't listen, I read, I can't relate to that experience. But where we are is really good.

Leo: I might get the book version of this one.

Steve: It is just - and again, I wasn't going to start until the number two was ready.

Leo: Right.

Steve: But when John said, yeah, I already finished it, and I'm reading it again, I thought, okay, I'm going to - I'm going to try it.

Leo: We're talking about, for those just joining us, Peter F. Hamilton, one of our favorite sci-fi authors, he writes these beautiful...

Steve: "The Archimedes Engine."

Leo: Yeah, and it's called "Exodus" is the first volume.

Steve: Is the first of two, a duology.

Leo: Yeah.

Steve: And, oh, wow. Okay. So we've got a great episode. Apple, who's a member of the CA/Browser Forum, has proposed that over time we bring maximum certificate life down to 45 days. To which I say, please, no, don't do it.

Leo: That's pretty quick.

Steve: Also, the SEC has fined four companies for downplaying the severity of the consequences of the SolarWinds attack on them, which is interesting. Google has added five new features, or will be - a couple are in beta - to Messenger, their Google Messenger app, including inappropriate content warnings, which is interesting because of course Apple did that a few years ago. And this brought me to an interesting question, and that is whether AI-driven local device-side filtering could be the resolution to the encryption dilemma forever. That is, solve the end-to-end encryption problem. Anyway, we'll talk about that.

Also, I tripped over, as a consequence of some news of them relocating, something I'd never been aware of before, a very nice-looking messenger app called Session, which is what you'd get if you were to marry Signal and Onion Routing from Tor.

Leo: Oh.

Steve: It's very interesting. I imagine our listeners are going to be jumping on this. Also just a quick look at the EU software liability moves. There were a couple other people produced some commentary that we did, we talked about that last week. We've got fake North Korean employees actually found to have been installing malware in some blockchain stuff. Also, answering a listener's question about whether he needed SpinRite to speed up an SSD, no, you don't. I'll touch on that. Also using ChatGPT to review and suggest improvements of code, another thought from a listener. And then I want to spend some time looking at the Internet's governance that has been trying to move the Internet to IPv6 for, yes, lo the past 25 years.

Leo: Yeah.

Steve: But the Internet just doesn't want to go. Why not? Will it ever? What's happened? The guy at APNIC, the Asia Pacific Network registry, has a really interesting take on the way the Internet has evolved such that, well, first of all, from some technology standpoints, we don't actually have the problem anymore that they were worried about needing IPv6 to solve. And why getting to places is no longer about addresses, it's about names. So, oh, and to cap all this off - as I said, we've got just a great podcast. But this Picture of the Week, OMG. It is - I just gave it the caption "There really are no words." This is one, it's a little cerebral, you've got to look at it for a minute and just think, oh, what the hell.

Leo: I haven't looked at it yet. It's on my screen. It's ready to be scrolled up into view. We will do that together in moments. And I can't wait. It's always fun. We should mention, if you want to get the show notes, easiest thing to do is go to Steve's site, GRC.com. Go to the podcast page, and every podcast has a very nice PDF of show notes that include that image. Or go to GRC.com/email and sign up for his newsletter. That way you can get it automatically ahead of time.

Steve: And Leo, I don't know if you looked at the timestamp on the email that you got. It was yesterday afternoon.

Leo: So Steve's wife is making him do this. I'm convinced. Does Lorrie say you've got to get this done so we can go out to dinner or something?

Steve: So by Sunday late morning I had finished the project I had been working on, which is the amalgamation of the ecommerce system and the new emailing system. I didn't have them communicating yet, and they had to so that we didn't have the databases desynchronized.

Leo: Right.

Steve: And so I thought, okay, I'm going to start working on the podcast. And as it turns out, it went well. There was lots of material. And by yesterday afternoon, Monday afternoon, I was done. And so I thought, I'm just going to give - actually it was 11,717 subscribers got it a day ahead.

Leo: A head start. Steve's been doing that more and more lately. So it's worth subscribing to get the early edition.

Steve: Yeah, you do. Well, and in fact I'm not going to step on it, but one of our listeners mentioned a benefit of that that hadn't occurred to me before.

Leo: Ah, okay.

Steve: So anyway, we've got a great podcast, and get ready for the Picture of the Week.

Leo: Now, I am going to - how should I do this - scroll up.

Steve: Yup.

Leo: First, before I show everybody else, I'm going to scroll up.

Steve: Consider what you see.

Leo: And the caption is "There really are no words." I see a fire truck. I see a fire hose - oh, dear. Okay. Okay. That's good. I'm showing you the picture now, everybody. Steve, do you want to describe it for our audio listeners?

Steve: So if you - oh. If you were a fire company, and you needed to have a hose go across an area where, you know, cars would be driving over it, you might - and we've seen this like with electrical cords that have to go across the floor. You put a protector around it, kind of like a little ramp up and down on either side so that, you know, you can roll over it. You won't get stuck on it. You won't squash it. You know, you'll protect it.

So we have that scenario here in today's Picture of the Week for Security Now! #998, where a fire hose is being protected with similar sort of little kind of ramps. The problem is they're not being protected from a car's tires rolling over the hose. They're being protected from a train. This is crossing a train track. And anybody, you know, who's

thought much about the way trains work, they have wheels that have flanges on the inside which are the things that keep the wheels on the track. And so the last thing you want to do is do anything to force those wheels up out of the grooves.

Leo: I think in all likelihood a train would just go right over the top of that and cut it in half.

Steve: I don't know.

Leo: You think it would cause a derailment?

Steve: Oh, oh, you mean slice it.

Leo: Slice it, right, yeah, because those wheels are sharp and just go [clicking].

Steve: I hope they're sharp.

Leo: You think it could derail it?

Steve: Because you'd want that - you would much rather have your hose cut than to have the train derailed. Which is the alternative.

Leo: You'd have another emergency to attend to very quickly. Unbelievable. Unbelievable.

Steve: To me this looks like maybe, like England? I don't know why I kind of have an English feeling to it. But, you know, like we don't have - it looks like there's a crossing gate, and the light is over - oh, I know why. It's aimed away from us, and it's on the right side.

Leo: On the left, yeah, yeah, yeah.

Steve: Meaning it would be on the left side if you were driving on the left side of the road.

Leo: Right, yeah.

Steve: So, and it looks like, do you see water coming out of the back left of the fire engine?

Leo: I see that, yeah. What is that all about?

Steve: We don't know what is going on. But it's not good, if any train is going to be coming down the tracks.

Leo: No.

Steve: Oh, goodness.

Leo: Oh, holy moly.

Steve: Anyway, one of our better pictures, I think. It's just - okay.

Leo: Say it with me. What could possibly go wrong?

Steve: Could possibly go wrong. Actually, that would have been a much better caption, Leo. That would have been a perfect caption for this picture.

Okay. So as our long-time listeners know, many years ago we spent many podcasts looking at the fiasco that was, and sadly still is, certificate revocation, noting that the system we had in place using CRLs - certificate revocation lists - was totally broken. And I put GRC's "revoked.grc.com" server online specifically for the purpose of vividly demonstrating the lie we were being told, that it just doesn't work.

Now, at the time, the OCSP solution - Online Certificate Status Protocol - seemed to be the best idea. But if users' browsers queried for OCSP status, like real-time, and the idea was that you could do it online, the browser could ask the CA is the certificate I've just received from the web server still good. The problem was it created both performance problems because of this extra need to do a query, and privacy issues because the CA would know everywhere that users were going based on their queries back to the CA's servers. So the solution to that was OCSP stapling, where the website's own server would make the OCSP query, thus no privacy concern there, and then, as the term was, "staple," meaning in some means electronically attach, these fresh OCSP results to the certificate that it was serving to the web browser so the web browser wouldn't have to go out and make a second request. Great solution.

But it seems that asking every web server in the world to do that was too high a bar to reach because, while some did, mine was, most weren't. So despite its promise and partial success, the CA/Browser Forum, which sets the industry's standards, recently decided - and we covered this a few, I guess about a month ago - to backtrack and return to the previous, the earlier user and formal endorsement of the earlier Certificate Revocation List system, which would move all of the website certificate checking to the browser. This had the benefit of allowing us to offer a terrific podcast explaining the technology of Bloom filters, which everyone enjoyed, and that technology very cleverly allows the user's browser to locally and very quickly determine the revocation status of any incoming certificates.

Okay. So that's where we are. Now, when you think about it, for a certificate to be valid, two things must be true. First of all, we must be between the certificate's "not valid before" and "not valid after" dates; and there must be no other indication that this thus otherwise valid certificate has nevertheless been revoked for some reason. Doesn't matter why. It's no longer good. So the conjunction of these two requirements means that the Certificate Revocation Lists, which are the things that will tell us if there's an

exception to the validity period test, they only need to cover any certificates that would otherwise be valid; right? This means that expired certificates will be automatically distrusted due to their expiration and can thereby be safely removed from the next update to the industry's Bloom filter-based CRL lists.

Okay. So if we want to keep the sizes of our Bloom filter CRLs down, shortening the lives of certificates is the way to do that. Or if, you know, if this still doesn't come to pass, because we've been at this for quite a while, and we've never gotten any form of revocation that actually works, so maybe just shortening is a good thing in general.

And this brings us to last week's news of a proposal by Apple, who is, as I mentioned at the top of the show, an active, very active member of the CA/Browser Forum, they're proposing to gradually reduce maximum web server certificate life from its current duration of 398 days, basically a year plus a month, all the way down to just 45 days. If this proposal were to be adopted, certificates would have their lives reduced in four steps, starting one year from now and ending in April of 2027.

It would go this way: We're currently at 398 days. That comfortable 398 would be cut nearly in half to 200 days, one year from now, in September of 2025. Actually a month ago, right, because we're ending October here in a second. Anyway, then a year later, in September 2026, it would be reduced by half again, from 200 to 100 days. And the final reduction would occur seven months later, in April of 2027, which would see web server certificate maximum lifespans reduced to just 45 days.

Okay, now, the only reason Let's Encrypt's 90-day certificate lifetimes are workable is through automation; right? So Apple must be assuming that by setting a clear schedule on the plan to decrease certificate lifespans, anyone who has not yet fully automated their server's certificate issuance with the ACME protocol - which is the standard that the industry has adopted for allowing a web server to automatically request a new certificate, anyone who hasn't already done that will be motivated to do so because, you know, the end is coming. You know, who wants to manually update their certificates with, you know, short of 45 days? Nobody.

So the problem is this creates some potential edge-case problems since it's not only web servers that depend upon TLS certificates. For example, I have one of personal interest that comes to mind. GRC, as we know, signs its outbound email using a mail server that's manually configured to use the same certificate files that are valid for the GRC.com domain. That's what you have to do to get DKIM working. And then DKIM and SPF, as we talked about recently, together allows you to obtain DMARC certification. And then the world believes that email coming from GRC actually did because it's signed.

Well, it's signed with the same certificate that DigiCert created for me for GRC's web server because it's from the GRC.com domain. At the moment, I only need to update the email's copy of those certificates annually. So it's manageable to do that through the email server's UI, which is the mechanism that provides for that. I don't know what would happen if I were to change the content of the files out from under the email server without it knowing, you know, using ACME-style updates. For all I know, it has private copies of the certificates which it might be holding open, you know, holding the files open to improve their speed of access, which would prevent them from being changed.

There's currently no programmatic way to inform the email server that it needs to change its certs since this has never been a problem or a necessity until now. Remember, once upon a time it was three years. And way back it was 10 years that we had certificate life. So, you know, it happens that I'm able to write code. So I could see that I might wind up having to add a new custom service to watch for my web server autonomously changing its certificates, then shut down the email server, update its copies of the certs, and restart it.

My point is, that's what's known as a royal effing kludge, and it is no way to run the world. And make no mistake, my email server is just a trivial example of the much larger problem on the horizon. Think of all the non-ACME-aware or non-ACME-capable, non-web server appliances we have today that have proliferated in the past decade and which now also need certificates of their own. What do they do? So, you know, perhaps this is the price we pay for progress. But I question, you know, this sort of brought to mind, I question why this should be imposed upon us and upon me. It's my certificate. It represents my domain of GRC.com. Why is it not also my choice how long that representation should be allowed to endure?

Okay, if I'm some big organization like Amazon.com, Bank of America, PayPal, where a great deal of damage could be done if a certificate got loose, I can see the problem. So such organizations ought to be given the option to shorten their certificates' lives in the interest of their own security. And in fact they can do that today. When I'm creating certificates at DigiCert, I'm prompted for the certificate's duration. 398 days is the maximum lifetime allowed. But there's no minimum. And DigiCert supports the ACME protocol, so automation for short-lived certificates is available from them. But why are short-lived certificates going to be imposed upon websites by the CA/Browser Forum and the industry's web browsers?

And let's get real here. As we know, revocation has never worked. Never. It's always been a feel-good fantasy. And the world didn't end when we only needed to reissue certificates once every three years with no effective ability to revoke them. Now the industry wants to radically reduce that to every six weeks? How are we not trying to solve a problem that doesn't actually exist, while at the same time creating a whole new range of new problems we've never had before? I'll bet there are myriad other instances, such as with my email server, where super-short-lived certificates will not be practical.

This sure seems like a mess being created without full consideration of its implications. Do these folks at the CA/Browser Forum fail to appreciate that web servers are no longer the only things that require TLS connections and the certificates that authenticate them and provide their privacy? And many of these devices that need certificates for a domain may not be able to run the ACME protocol because they are DV (Domain Validation) certs.

I dropped my use of EV certificates because that became wasted money once browsers no longer awarded those websites using EV certificates with any special UI treatment. You didn't get a little green glow up there in the UI bar. But I've continued using OV, those are Organization Validation certificates, since they're one notch up from the lowest form of certificate, the Domain Validation DV cert, which Let's Encrypt uses because that's all it's doing is just validating, yes, you're in control of that domain.

But if we're all forced to automate certificate issuance, I can't see any reason then why everyone won't be pushed down to that lowest common denominator of Domain Validation certificates, the issuance of which Let's Encrypt has successfully automated. At that point, certificates all become free, and today's Certificate Authorities lose a huge chunk of their recurring business. How is that good for them? And the fact is, simple Domain Validation provides a lesser level of assurance than Organization Validation. So how is forcing everyone down to that lowest common denominator good for the overall security of the world?

I suppose that Apple, with their entirely closed ecosystem, may see some advantage to this. So, fine. They're welcome to have whatever super-short-lived certificates they want for their own domains. But more than anything, I'm left wondering why the lifetime of the certificates I use to validate the validity of my own domain, in all of its various applications - web, email, and so forth - why that's not my business, and why that's not being left up to me?

Leo: So if it were Google saying this, I might worry because Google has this power to enforce its verkakte plans all the time. But it's Apple. Who cares? Is there any chance that this is going to become the rule?

Steve: Yes. Remember that Apple, when we went to 398 days, they said they would dishonor any certificate that had a longer life.

Leo: Because they have Safari; right.

Steve: Well, exactly. All of their iDevices, the certificate has both a "not valid before" and a "not valid after." So if those two dates are further than 398 days apart, Apple just says, sorry, this is an invalid certificate.

Leo: But they're not going to unilaterally impose a 45-day limit. They would want the CA/Browser Forum to agree; right?

Steve: Yes. And so that's what's happened is that there's a thread discussing this which is suggesting this timeline for bringing it down to 45 days. And I just - I do not see the logic in that. I see huge downside consequences.

Leo: A lot of downside.

Steve: And why is it any of their business how long I want my certificate to assert GRC.com for? I will take responsibility for that. It's in an HSM. It is safe. It cannot be stolen. And revocation, you know, maybe it's going to be coming back with Bloom filters, we hope. So if the worst happened, we could still revoke. But the idea that, like, I won't be able to purchase a trusted certificate from a CA longer than 45 days, that's not a good place.

Leo: What is the rationale? Why does Apple want to make it so short?

Steve: It can only be so that you are constantly having to reassert your control over the domain and the certificate.

Leo: So for the health of the Internet, then.

Steve: Yes. And do you see a big problem there? There is no big problem there. They used to be for three years, and everything was just fine. We're still here.

Leo: Right.

Steve: And we never had revocation that worked. It wasn't a problem.

Leo: So they're solving a problem that doesn't exist with a solution that causes many more problems.

Steve: I think they're going to end up, people are just going to say no.

Leo: I hope so.

Steve: Because we've just come down from three years to one year. And, you know, and at the time I said the only good thing I could see about this, Leo, is that every three years I'd forgotten how to do it. You know?

Leo: Right, right.

Steve: There was so much time in between, it was like, you know, I have to run this through SSL, I mean, oh, yeah.

Leo: That's when you had to do it manually.

Steve: OpenSSL with a certain weird command line to get a PFX format and then blah blah. And every three years I'd forgotten.

Leo: And then CRL and, yeah.

Steve: Now every year it's like, oh, yeah, okay, I've got to do this again. But so it has the advantage of - oh, and you know, of course, how many times have we seen websites where they're like, whoops, our certificate expired. Because it was, you know, three years ago Paul worked here. Well, Paul's no longer here, and he was the guy who did the certificates.

Leo: Right, right. But we still see that. Let's Encrypt has had real success with the scripted re-update.

Steve: Oh, my god, they've taken over the TLS market.

Leo: Right.

Steve: It's like three quarters or two thirds of all certificates because nobody wants to pay. It's like, wait, I can get it for free?

Leo: It's free, and the script runs automatically. You don't even have to think about it, and you don't have to worry about Paul anymore because you just re-up every, what is it, 90 days?

Steve: It's 90 days for Let's Encrypt; right.

Leo: I don't see any reason, though, to make it half that. That's crazy.

Steve: It is. I don't see it either. And remember, you can still get a certificate, even if you're using Let's Encrypt for your web browsers, you can still buy a one-year cert for other things. And so, like, so all the appliances that we have that want to do TLS connections, you could still purchase a longer-lived certificate. So, you know, and when I was thinking this through, one possibility would be to allow non-web certs to have a longer life, where because in every cert certificate...

Leo: There you go, yeah.

Steve: ...it does state what the uses of the cert are. So automatable reissuance could have a shorter life, but then it doesn't solve the problem because, if what you're worried about is the certificate being stolen, apparently they're worried about anything with longer than 45 days being anywhere. It's like, I just - I do not understand. And again, why is it their business? This is we had three years, we had no problems except, you know, Paul leaving the company.

Leo: Did Paul leave the company?

Steve: And we're half an hour in, Leo. Let's take a break.

Leo: Oh, all right.

Steve: And then we'll talk about the SEC levying fines against four companies who lied.

Leo: Oh. Shame on them.

Steve: You don't want to do that to the SEC.

Leo: How dare they, those lying liars. On we go, Mr. G.

Steve: Okay. So one of the rules of the road is that companies that are owned by the public through publicly traded stock have a fiduciary duty to tell the truth to their stockholders...

Leo: Oh, wouldn't that be nice.

Steve: Yes, when something occurs that could meaningfully affect the company's value.

Leo: Yes.

Steve: For example, on December 14th, 2020, the day after The Washington Post reported that multiple government agencies had been breached through SolarWinds Orion software, the company itself, SolarWinds, stated in an SEC filing that fewer than 18,000 of its 33,000 Orion customers were affected. Still, 18,000 customers affected made it a monumental breach. And there was plenty of fault to be found in SolarWinds' previous and subsequent behavior. But they fessed up. They said, okay, this is what happened.

But I didn't bring this up to talk about them. I wanted to share some interesting reporting by CyberScoop whose headline a week ago was: "SEC hits four companies with fines for misleading disclosures about SolarWinds hack." In other words, for misleading the public about the impact their use of SolarWinds' Orion software had on their businesses, and how it might affect, you know, their shareholders' value.

CyberScoop's subhead was: "Unisys, Avaya, Checkpoint, and Mimecast will pay fines to settle charges that they downplayed in SEC filings the extent of the compromise." And this is the point that I wanted to make. The management of companies owned by the public need to tell the truth. So let's take a closer look at this.

CyberScoop wrote: "The Securities and Exchange Commission (SEC) said it has reached a settlement with four companies for making materially misleading statements about the impact of the 2020 SolarWinds Orion software breach on their businesses. The regulator charged the four companies Unisys, Avaya Holdings Corp., Checkpoint Software Technologies, and Mimecast Limited with 'minimizing the compromise or describing the damage to internal systems and data as theoretical, despite knowing that substantial amounts of information had been stolen.'" In other words, they outright lied to their shareholders.

CyberScoop said: "The acting director of the SEC's Division of Enforcement said in a statement: 'As today's enforcement actions reflect, while public companies may become targets of cyberattacks, it's incumbent upon them to not further victimize their shareholders or other members of the investing public by providing misleading disclosures about the cybersecurity incidents they've encountered. Here, the SEC's orders find that these companies provided misleading disclosures about the incidents at issue, leaving investors in the dark about the true scope of those incidents.'

"As part of the settlement agreement reached, the companies have agreed to pay fines with no admission of wrongdoing." Okay, so in the first place they're not having to say "we lied." Okay, so there's that. But then I was unimpressed, frankly, by the amounts. Unisys will pay $4 million, Avaya $1 million, Checkpoint $995,000, and Mimecast $990,000.

"According to the SEC, by December 2020 Avaya, for example, already knew that at least one cloud server holding customer data and another server for their lab network had both been breached by hackers working for the Russian government. Later that month, a third-party service provider alerted the company that its cloud email and file-sharing systems had also been breached, likely by the same group and through means other than the SolarWinds Orion software.

"A follow-up investigation identified more than 145 shared files accessed by the threat actor, along with evidence that the Russian group known as APT29, a.k.a. Cozy Bear, monitored the emails of the company's cybersecurity incident responders." So they were deeply penetrated, and they knew it. "Despite this, in a February 2021 quarterly report a couple months later, Avaya described the impact in far more muted terms, saying the

evidence showed the threat actors accessed only 'a limited number of company email messages'" - okay, that's a little gray - "and there was 'no current evidence of unauthorized access in our other internal systems.'" Okay, so you can call into question the word "current"; right? They knew that those representations were flatly false.

"Unisys's investigation uncovered that, following the disclosure of a device running Orion, multiple systems seven network and 34 cloud-based accounts, including some with admin privileges were accessed over the course of 16 months. The threat actors also repeatedly connected to their network and transferred more than 33GB of data. But the SEC's cease-and-desist order stated that Unisys had 'inaccurately described the existence of successful intrusions and the risk of unauthorized access to data and information in hypothetical terms, despite knowing that intrusions had actually happened and, in fact, involved unauthorized access and exfiltration of confidential and/or proprietary information.' The company also appeared to have no formal procedures in place for identifying and communicating high-risk breaches to executive leadership for disclosure."

Anyway, and there are similar instances at Checkpoint and Mimecast. The problem here is I'd like to be able to draw a clear moral to this story, as it sort of started out seeming. But given the extremely modest size of the settlements relative to each company's revenue, it's not at all clear to me that the moral of our story here is that they should have divulged more. During the heat of the moment, the short-term impact upon their stock price may have been more severe than these fines. And coming four years after the event, it's reduced to a shrug.

So I doubt that this outcome will wind up teaching any other companies any important lessons. And any companies that did the right thing at the time and were then punished by their stockholders for telling the truth might actually take away the opposite lesson: Let's just lie, sweep it all under the rug for now. And then, if three or four years later, you know, after we're hit with a modest tax for having done that, you know, the world will have moved on anyway. We'll happily pay it, and we'll have lost less money than if we had told the truth right up front. I mean, that's the takeaway from this.

Leo: The cost of doing business, these companies always say.

Steve: Yup.

Leo: Yup. The cost of lying to our stockholders. Wow.

Steve: Okay. So last Tuesday, Google's Security Blog posted the news of five new protections being added to their Google Messages app. Although Google's postings are often a bit too full of marketing hype for my own taste, I thought this one would be worth sharing, and it's not too long. So Google wrote, and here's the marketing intro: "Every day over a billion people use Google Messages to communicate. That's why we've made security a top priority, building in powerful on-device, AI-powered filters and advanced security that protects users from two billion suspicious messages a month. With end-to-end encrypted RCS conversations, you can communicate privately with other Google Messages RCS users. And we're not stopping there. We're committed to constantly developing new controls and features to make your conversations on Google Messages even more secure and private.

"As part of cybersecurity awareness month" - they're getting it in just before Halloween, when it ends - "we're sharing five new protections to help keep you safe when using Google Messages on Android." Okay. So we've got: "Enhanced detection protects you

from package delivery and job scams." And I'm going to skip the paragraph describing it because we all know what it means, you know, they're looking at the messages coming in, and they're going to do some filtering to recognize when this is basically spam and, you know, flag it, warn you, whatever.

Number two: "Intelligent warnings alert you about potentially dangerous links." Same thing there. They're getting into your encrypted massaging using device-side AI-powered filters to deal with that. But this one's short. They said: "In the past year, we've been piloting more protections for Google Messages users when they receive text messages with potentially dangerous links." Again, incoming text messages being examined. They said: "In India, Thailand, Malaysia, and Singapore, Google Messages warns users when they get a link from unknown senders and blocks messages with links from suspicious senders. We're in the process of expanding this feature globally later this year." So there's not much of this year left, so that'll be coming soon to Google Messages for everybody else.

"Controls to turn off messages from unknown international senders," another benefit. But it was number four that most caught my attention: "Sensitive Content Warnings give you control over seeing and sending images that may contain nudity." They said: "At Google, we aim to provide users with a variety of ways to protect themselves against unwanted content, while keeping them in control of their data. This is why we're introducing Sensitive Content Warnings for Google Messages. Sensitive Content Warnings is an optional feature that blurs images that may contain nudity before viewing, and then prompts with a 'speed bump'" - is the way they phrased it - "that contains help-finding resources and options, including to view the content.

"When the feature is enabled, and an image that may contain nudity is about to be sent or forwarded, it also provides a speed bump to remind users of the risks of sending nude imagery and preventing accidental shares. All of this happens on-device to protect your privacy and keep end-to-end encrypted message content private to only sender and recipient. Sensitive Content Warnings does not allow Google access to the content of your images, nor does Google know that nudity may have been detected. This feature is opt-in for adults, managed via Android Settings, and is opt-out for users under 18 years of age." In other words, on by default for kids. "Sensitive Content Warnings will be rolled out to Android 9+ devices, including Android Go devices, with Google Messages in the coming months." So I'll get back to that in a second.

The last piece of the five was "More confirmation about who you're messaging." Basically they're allowing an out-of-band explicit public key verification for messaging, which other messengers, notably Threema was a leader in this pack, who were doing, you know, traditional standard-style cryptography where we knew what a public key was and, you know, verifying somebody's shared public key was useful. So that's the fifth thing.

Okay. But I want to skip back, as I said, to that fourth new feature, Sensitive Content Warnings. Apple announced their Sensitive Content Warnings in iOS 15 where the smartphone would detect probably-sensitive content and warn its user before displaying it. Despite that potentially privacy-invading feature, which has now been in place for several years, we're all still here, just like we are without certificate revocation. The world did not end. Not only did it not end, you know, when smartphones began looking at what their users were doing, it didn't even slow down. So the idea of device-side image recognition and detection has not proven to be a problem, and Google has clearly decided to follow. But I believe there may be a larger story here. I suspect that this will be the way the world ultimately resolves that thorny end-to-end encryption dilemma that we've been looking at for several years now.

As we know, Apple initially really stepped in it by telling people that their phones would be pre-loaded with an exhaustive library of the world's most horrible known CSAM, you

know, Child Sexual Abuse Material. No one wanted anything to do with having that crap on their phone, and even explaining that it would be fuzzy matching hashes rather than actual images did nothing to mollify those who said, "Uh, Apple, thanks anyway. I'll go get an Android phone before I let you put anything like that on my iPhone."

Apple received that message loud and clear and quickly dropped the effort. But then, right in the middle of the various European governments, and especially the UK's very public struggles over this issue, facing serious pushback from every encrypted messaging vendor, saying they would rather leave than compromise their users' security, AI suddenly emerges on the scene and pretty much blows everyone's mind with its capabilities and with what it means for the world's future.

If there's been any explicit mention of what AI might mean to highly effective, local, on-device filtering of personal messaging content, I've missed it. But the application seems utterly obvious, and I think this solves the problem in a way that everyone can feel quite comfortable with. The politicians get to tell their constituents that "next-generation AI" will be watching everything their kids' smartphones send and receive and will be able to take whatever actions are necessary without ever needing to interfere with or break any of the protections provided by full, true, end-to-end encryption. So everyone retains their privacy with full encryption, and the bad guys will soon learn that they're no longer able to use any off-the-shelf smartphones to send or receive that crap. It seems to me this really does put that particular intractable problem to rest, and just in the nick of time.

I'll note one more thing about this. It's foreseeable that the behavior recognition provided by AI-based on-device filtering will eventually and probably inevitably be extended to encompass additional unlawful behavior. We know that governments and their intelligence agencies have been credibly arguing that terrorists are using impenetrable encryption to organize their criminal activities. So I would not be surprised if future AI-driven device-side detection were not further expanded to encompass more than just the protection of children. This, of course, raises the specter of Big Brother, you know, monitoring our behavior and profiling, which is creepy all by itself. And I'm not suggesting that's an entirely good thing because it does create a slippery slope. But at least there we can apply some calibration and implement whatever policies we choose as a society.

What is an entirely good thing is that those governments and their intelligence agencies who have been insisting that breaking encryption and monitoring their population is the only way to be safe will have had those arguments short-circuited by AI. Those arguments will finally be put to rest with encryption having survived intact and, arguably, giving the intelligence agencies what they need. So anyway, I just - it hadn't occurred to me, Leo, before now. But it seems to me that that's a powerful thing that AI on the device can do. And it really can and should satisfy everybody.

Leo: Yeah. Well, I'm glad Google's doing what they're doing. I mean, I think that seems like a sensible plan.

Steve: Yup, yup.

Leo: And as you note, it's opt-in for adults and opt-out for kids, which is exactly how it should be.

Steve: Yup, agreed. Okay. I stumbled on a surprising app that I was never aware of. So while we're on the subject of encrypted apps, an app known as Session - and anybody

who's listening live and wants to jump ahead, GetSession.org is the URL. An app known as Session is a small, but increasingly popular, encrypted messaging app. Session announced that it would be moving its operations outside of Australia - get this, Leo - after the country's federal law enforcement agency visited an employee's residence and asked them questions about the app and about a particular user of the app. As a result of that nighttime intrusion, Session will henceforth be maintained by a neutral organization based in Switzerland.

Leo: Good.

Steve: Yes.

Leo: That's appalling.

Steve: It is. It was like, whoa. You know, a knock on your door, and there's, you know, Australian federal law enforcement saying, uh, you work for this Sessions company; right? So we need some information about one of your users.

Leo: God.

Steve: Well, wait till you hear how impossible it is for them to answer that question. 404 Media noted that this move signals the increasing pressure on maintainers of encrypted messaging apps, both when it comes to governments seeking more data on app users, as well as targeting messaging app companies themselves. They cited the recent arrest of Telegram's CEO in France last August. Alex Linton, the president of the newly formed Session Technology Foundation, which will publish the Session app from Switzerland, told 404 Media in a statement: "Ultimately, we were given the choice between remaining in Australia or relocating to a more privacy-friendly jurisdiction, such as Switzerland. The app will still function in Australia, but we won't."

Okay. So I wasn't aware of the Session messaging app at all until I picked up on the news of this departure. But it looks quite interesting, and I wanted to put it on everyone's radar. It appears to be what you would get if you were to combine the ultra-robust and well-proven Signal protocol, which Session forked on GitHub, with the distributed IP-hiding Tor-style onion routing which we briefly discussed again recently. And on top of all that, Session is 100% open source; and, as I mentioned, all of it lives on GitHub.

Since all of this piqued my curiosity, I tracked down a recent white paper describing Session, which was written in July of this year. It's titled: "Session: End-to-End Encrypted Conversations With Minimal Metadata Leakage." And that's the key. The white paper's Abstract described Session in a couple sentences. It says: "Session is an open-source, public-key-based secure messaging application which uses a set of decentralized storage servers and an onion routing protocol to send end-to-end encrypted messages with minimal exposure of user metadata. It does this while providing the common features expected of mainstream messaging applications, such as multi-device syncing, offline inboxes, and voice/video calling."

Huh. Okay. Well, I would imagine that the Australian Feds were probably left quite unsatisfied by the answers anyone knowledgeable of Session's design would have provided to them during their visit in the evening. They would have explained that

Session's messaging transport was deliberately designed like Tor's to hide each endpoint's IP address through a multi-hop globally distributed server network, and that the entire content of the messages used the impenetrable Signal protocol used by Signal and WhatsApp to exchange authenticated messages between the parties.

And if this didn't already sound wonderful, listen to the system's mission statement from the white paper's introduction. They said: "Over the past 10 years, there's been a significant increase in the usage of instant messengers, with the most widely used messengers each having amassed over one billion users. The potential privacy and security shortfalls of many popular messaging applications have been widely discussed. Most current methods of protecting user data are focused on encrypting the contents of messages, an approach which has been relatively successful. The widespread deployment of end-to-end encryption does increase user privacy; however, it largely fails to address the growing use of metadata by corporate and state-level actors as a method of tracking user activity.

"In the context of private messaging, metadata can include the IP addresses and phone numbers of the participants, the time and quantity of sent messages, and the relationship each account has with other accounts. Increasingly, it is the existence and analysis of this metadata that poses a significant privacy risk to journalists, protesters, and human rights activists. Session is, in large part, a response to this growing risk. It provides robust metadata protection on top of existing cryptographic protocols which have already been proven to be effective in providing secure communication channels.

"Session reduces metadata collection in three key ways. First, Session does not require users to provide a phone number, email address, or any other similar identifier when registering a new account." In fact, no identifier. I've done it. "Instead, pseudonymous public-private key pairs are the basis of an account's identity, and the sole basis. Secondly, Session makes it difficult to link IP addresses to accounts or messages sent or received by users, through the use of an onion routing protocol." Same thing Tor does.

"And third, Session does not rely on central servers. A decentralized network of thousands of economically incentivized nodes performs all core messaging functionality. For those services where decentralization is impractical, like storage of large attachments and hosting of large group chat channels, Session allows users to self-host infrastructure and rely on built-in encryption and metadata protection to mitigate trust concerns."

In other words, wow. As we know, Pavel Durov, the Telegram guy, freed himself by agreeing to, where warranted, share IP addresses and whatever other metadata Telegram collected with law enforcement. And we know that Apple, Signal, and WhatsApp all similarly keep themselves out of hot water with governments and law enforcement by cooperating to the degree they're able to. And they are able to. They're able to provide IP addresses and related-party identifiers. They may not be able to peer into the content of conversations, but the fact of those conversations and the identity of the parties conversing is knowable, shared, and sharable.

And it occurred to me, since I put this down, another perfect example of the power of metadata is cryptocurrency and the blockchain. Much was made of the fact that, oh, it's completely anonymous. Don't worry about this. It's just a little - it's just, you know, you have your key in the blockchain. All transactions are anonymous. Well, we know how well that worked; right? We're able to see money moving and perform associations when it comes out of the cryptocurrency realm. So again, we're not able to see who, but there's metadata that's been left behind. Session was created to, to every degree possible, also prevent metadata leakage.

So I suppose we should not be surprised that the guys who married the Signal messaging protocol with Tor's onion routing to deliberately create a hyper-private

messaging system saw the clear handwriting on the wall and decided - after that visit from their local feds - that they would need to move from Australia sooner or later, so it might as well be sooner.

The messaging app, again, is called "Session," and it's available in several flavors for Android and iOS smartphones, as well as for Windows, Mac, and Linux desktops. From here it appears to be a total win. Establishing an anonymous identity with a public-private key pair is exactly the right way to go, and that's exactly what they do, plus much more, and with all their source code being openly managed on GitHub.

In addition to the 34-page technical white paper, there's also a highly accessible five-page "light paper," as they called it, which carries their slogan "Send messages, not metadata." So the URL, once again, is GetSession.org, where you'll find a software download page (GetSession.org/download) as well as links to both that five-page light paper and the full 34-page white paper. So it looks like a complete win to me.

Leo: So I have a couple of questions for you about this, prompted by some folks in our YouTube chat. We have chat now in eight different platforms. So I'm trying to monitor it all, but...

Steve: Are they all merged?

Leo: I have a merged view, so I can see it.

Steve: Wow.

Leo: Imran in our YouTube chat says "Note that Session has removed perfect forward secrecy and deniability from the Signal protocol." And they did that a few years ago. They say you don't need PFS because that would require full access to your device. And if you had full access, you know, really the jig is up no matter what.

Steve: Yeah.

Leo: And deniability is not necessary because they don't keep any metadata about you so that you don't have to worry about that. Does that seem true to you?

Steve: I think that's correct. The concern with perfect forward secrecy is that the NSA is filling that large server farm...

Leo: They're saving it all.

Steve: Yes, that massive data warehouse. And at some point in the future, if and when we actually do get quantum computing able to break today's current public key technology, then they can retroactively go back and crack that. Unfortunately, or fortunately, rather, Signal has already gone to post-quantum technology. So again, the concern is that, if you're not - so perfect forward secrecy, if you're constantly rekeying,

cuts off somebody who manages to penetrate public key technology for the duration of your use of that key. That allows them to get the symmetric key and decrypt that chunk of conversation. It's not clear at all today whether there's ever going to be a way to do that for anybody using the Signal protocol where you're using both pre- and post-quantum public key technology.

Leo: So it doesn't really matter.

Steve: They needed to do that in order to add the features that they wanted to.

Leo: Right, right, that makes sense. Okay.

Steve: I think that, you know, the idea...

Leo: I mean, the only real disadvantage is that you'll be in the only one in your family using it.

Steve: Yes, yes. And so it would be where you have a situation where you have some specific people that you want to have a really guaranteed private conversation with.

Leo: Right.

Steve: You know, it's not going to be like, oh, you know, what's your Signal handle and then, you know, add them to Signal. But still, for somebody who really wants, you know, true private communications, this goes further now than anything we've seen so far. They've got the state-of-the-art best messaging encryption technology in Signal, married to onion routing.

Leo: Which is quite clever, and no server has the full message. That's the interesting thing; right?

Steve: Right.

Leo: Yeah.

Steve: Right, it's completely decentralized.

Leo: I think that's so cool. And it always bothered me that Signal wanted my phone number. I just - it didn't feel like that was [crosstalk].

Steve: Yeah, and I think they've announced they're backing away from that now; right?

Leo: Yeah, they have usernames. They keep saying that, yeah.

Steve: Yeah. And it's like, okay, you know, and will it be many-to-many? I know that I was able to have it on one phone and one desktop. But I wasn't able to have it on two phones and multiple desktops. So, you know, some arbitrary limitations. I downloaded this thing, and it's on all of my phones and desktops, and they found each other.

Leo: With the same private key?

Steve: Yes.

Leo: So you're able to propagate it to other devices.

Steve: Yes.

Leo: Oh, that's cool. That's cool.

Steve: Yes. When you create it, it gives you a QR code. It shows it to you in hex and in that word salad form. You know, bunny, gopher, lawn...

Leo: It's memorable yeah, yeah.

Steve: ...artichoke, asparagus.

Leo: Right.

Steve: And so I copied that and then pasted that into a different device, and it found me and said, oh, here you are. And it got my picture, and everything worked.

Leo: I'm going to install it right now.

Steve: It's slick. And again, all three desktop platforms - Windows, Mac, and Linux - and both phone platforms.

Leo: Nice.

Steve: Let's take another break. And then we're going to revisit software liability briefly, and then close the loop, and plow into this question of the future of the Internet and does anyone even care about IPv6 anymore?

Leo: Wow. You know, we used to have Vint Cerf on, and he was like, banging the drum. We're going to run out of IP addresses. We're going to run out of IP addresses. We've got to go to IPv6.

Steve: What's really interesting is a chart of the U.S. adoption. But we'll get there.

Leo: Oh, I can't wait. Steven, your turn.

Steve: So, thank you. The EU's proposed wholesale revision of the software liability issue has, not surprisingly, drawn a huge amount of attention from the tech press. We gave it enough attention here last week, but I was glad to see that I didn't misstate or misinterpret the effect and intent of this new EU Directive. It really is what it appears to be. One reporter about this wrote: "The EU and U.S. are taking very different approaches to the introduction of liability for software products. While the U.S. kicks the can down the road, the EU is rolling a hand grenade down it to see what happens.

"Under the status quo, the software industry is extensively protected from liability for defects or issues, and this results in" - I love this - "systemic underinvestment in product security. Authorities believe that by making software companies liable for damages when they peddle crapware, those companies will be motivated to improve product security." And of course we can only hope.

I also wanted to share part of what another writer wrote for The Record. He wrote: "Six years after Congress tasked a group of cybersecurity experts" - U.S. Congress - "with reimagining America's approach to digital security" - get this - "virtually all of that group's proposals have been implemented. But there's one glaring exception that has especially bedeviled policymakers and advocates, a proposal to make software companies legally liable for major failures caused by flawed code.

"Software liability," he writes, "was a landmark recommendation of the Cyberspace Solarium Commission, a bipartisan team of lawmakers and outside experts that dramatically elevated the government's attention to cyber policy through an influential report that has seen roughly 80% of its 82 recommendations adopted. Recent hacks and outages including at leading vendors like Microsoft and CrowdStrike have demonstrated the urgent need to hold software companies accountable, according to advocates for software liability standards.

"But despite the Solarium Commission's high-profile backing and the avowed interest of the Biden administration, this long-discussed idea has not borne fruit. Interviews with legal experts, technologists, and tech-industry representatives reveal why: Software liability is extremely difficult to design, with multiple competing approaches; and the industry warns that it will wreck innovation and even undermine security. Jim Dempsey, senior policy adviser at Stanford University's Program on Geopolitics, Technology, and Governance said: 'The Solarium Commission and Congress knew that this was going to be a multiyear effort to get this done. This is a very, very, very hard problem.

"'A recent spate of massive cyberattacks and global disruptions including the SolarWinds supply-chain attack, the MOVEit ransomware campaign, the Ivanti hacks, the CrowdStrike outage, and Microsoft's parade of breaches has shined a spotlight on the world's vulnerability to widely distributed, but sometimes poorly written, code.' Dempsey added: 'There's a widespread recognition that something's got to change. We're way too heavily dependent on software that has way too many vulnerabilities.'

"The software industry's repeated failures have exasperated experts who see little urgency to address the roots of the problem, but bringing companies to heel will be extremely difficult. An associate professor at Fordham School of Law who specializes in cybersecurity and platform liability said: 'We have literally protected software from almost all forms of liability, comprehensively, since the inception of the industry decades ago. It's just a golden-child industry.' Virtually all software licenses contain clauses immunizing vendors from liability. Policymakers originally accepted this practice as the cost of helping a nascent industry flourish. But now that the industry is mature, and its products power all kinds of critical services, it will be an uphill battle to untangle what Dempsey called the 'intersecting legal doctrines that have insulated software developers from the consequences of the flaws in their products.'"

So Leo, we, this podcast, certainly have not been alone in, like, just observing over the last 20 years that we've been doing this, like, this is wrong. This has to change. But also, change is hard. In other words, IPv6. No.

Okay. One last little point that I thought was interesting. As we know, a recurring event in security news recently has been the industry's inadvertent hiring of fake IT workers, generally those purporting to come, well, purporting to be domestic, but actually it turns out working and working for North Korea, or at least North Korean interests. Hopefully this has not been happening for long, you know, undetected, since there really seems to be a lot of it going around. Maybe we're just suddenly you know, shining a light on it, so we're seeing a lot of it. I shared the hoops that I had to jump through recently during that one-way video conference with a DigiCert agent, following his instructions as I moved my hands around my face, holding up the government-issued ID card and demonstrating that I was me.

As far as I know, the coverage of this has not actually revealed, that is, the coverage of the North Korean identity spoofing hasn't actually revealed any malfeasance on the part of these North Korean employees before now. It is certainly illegal to hire them, but they were faking their identities. It turns out that's changed. The creator of a blockchain known as Cosmos, the Cosmos blockchain, has admitted that the company inadvertently hired a North Korean IT worker. The company said the FBI notified the project about the North Korean worker; but, get this, the individual at the company who received the notification did not report the incident to his managers. What? And moreover, Cosmos says that the code contributed by the North Korean worker did contain at least one major vulnerability. The company is now performing a security audit to review all the code for other issues.

So we can only hope that these now continuing revelations will lead to many more real-time video conferences such as the one that I had with DigiCert to prove that I was actually me. You know, just sending, you know, forwarding a file with some headshots, that's not going to do it any longer.

Oh. And I mentioned this at the top. A listener suggested something I hadn't thought of before. His name is Brian. He said: "Please add me to your Security Now! podcast GRC list." He said: "I'm an occasional listener and appreciate all of your information and tips shared. Regards, Brian." That's all he said, but I wanted to share this because Brian is a mode of listener who can obtain a value from GRC's weekly podcast synopsis mailing that I had never considered. I often do hear from listeners who have fallen behind in listening, or who aren't always able to find the time to listen.

So Brian's note made me realize that the weekly mailings which, as I said at the top of the show went out to 11,717 people yesterday afternoon, in this case, can come in quite handy when making a determination about how to invest one's time. You look at the list, you go, ooh, there are a couple things here that I want to hear about. And then you grab the podcast. So thank you, Brian, for the idea.

Leo: Yeah, it's good for us to remember there are people who don't listen to every single show.

Steve: Yes.

Leo: I think that's an excellent point, yeah.

Steve: In this day and age there is a lot of competition for people's time and attention.

Leo: For sure. It always worried me, you know, that people would give up on the show if they couldn't listen to every one. You know. I stopped subscribing to The New Yorker because it ends up being such a big pile of magazines, and there's this guilt, like you have to read every issue instead of just dipping into it. So don't feel guilty. It's okay to not listen to every episode.

Steve: Right. And if you subscribe to the GRC list...

Leo: You'll know what you're missing.

Steve: You know, just go to GRC.com/mail and sign up. Add yourself to the Security Now! list. We have two. One's only for Security Now! listeners, and the other is just general GRC news. I'm sure, since I'll be talking about anything I'm doing at GRC - oh, speaking of which, I forgot to mention that because I finished the podcast yesterday afternoon, yesterday evening I updated GRC's technology for four digits of podcast numbering. So when we go from 999 to 1000, everything should work smoothly. So that is now in place.

So, okay. Two more pieces. Martin in Denmark said: "Hi, Steve. Love the podcast, been with you guys from Episode 0." Unfortunately, I do wish we'd started at zero, Leo. It just didn't occur to me. You know, we were green back then.

Leo: What did we know?

Steve: We were newbies. I thought we're never going to get to 999. We're not even getting to 300. So here we are.

Leo: I just want to point out, as a coder, there's a language that I love in every respect called Julia. My biggest complaint is it counts arrays from one, not zero. And I feel like, I'm sorry, I just - I can't do that. I just can't do that. In every other respect it's a wonderful language. But that, that's a bridge too far.

Steve: Yeah. It's supposed to be an offset, not a number.

Leo: Exactly. Zero or one.

Steve: And so is it a number or an offset?

Leo: Right, right. Oh, well.

Steve: So Martin in Denmark says: "I have a question about the stuff SpinRite does when 'speeding up an SSD.'" He said: "My computer is due for a reformat and a reinstall of Windows. Windows is slowing down, as it does, but it seems worse than 'usual,'" he has in quotes, "so I think my SSD could use a little help. Since I'm going to nuke the drive anyway, is there a way to do the same stuff that SpinRite does without SpinRite?" He said: "I assume that using Windows installer or diskpart to 'clean' the disk just wipes the filesystem/partition table and does nothing else. Am I right that a poor man's solution would be to delete the partitions on the drive and make a new one, and then fill it with random data? I don't own SpinRite," he said, "(money reasons yada yada) and was just wondering if there is another way, as I don't care about the data on the drive. Regards, Martin in Denmark."

So here's what I wrote in reply to Martin's email, which was written to me at securitynow@grc.com, which is the way anybody who is registered with GRC's email system is able to send me email, like I just read. I wrote: "Hi, Martin. You don't need SpinRite for that at all. The only magic SpinRite does, aside from perhaps helping hugely to recover from any trouble encountered in the process, is rewriting the data on an SSD. But it's the writing, not the rewriting, that's the key here. So if you're going to be reinstalling Windows, that act of re-installation will inherently be overwriting, and thus writing, which is the goal."

And I said: "We've discovered that SSDs can grow surprisingly slow without otherwise complaining as the years go by without regions of their media ever being rewritten. SpinRite makes refreshing SSDs with data in place easy. But if retaining an SSD's current data is not needed, then neither is SpinRite. A standard reinstallation of Windows will entirely do the trick for you." So just a heads-up for anybody else who may be in Martin's situation. You know, I'm happy to share that. We're seeing, like, example after example of people saying, OMG, I can't believe how much faster my laptop is after I ran SpinRite over it. So there it's certainly easier to do that in a couple hours than reinstall Windows. But Martin wants to do it anyway.

Oh, I forgot to mention that he got the show notes yesterday afternoon. He saw my reply in the show notes, and he wrote back and said: "Just wanted to let you know I reinstalled Windows, and oh my god, is it faster." He said: "It was more faster than I expected it to be." So indeed.

Leo: He's in the YouTube. He's watching on YouTube right now. He's in the chat. He said: "That's my question! Yay!"

Steve: What time is it in Denmark right now?

Leo: Oh, man, yeah, it's pretty late. Almost midnight.

Steve: Okay. And our last bit of feedback, Alain Gyger. Oh, and Leo, he's a fellow ham. K6ACG is his call sign. He said: "Hi, Steve. I'm really liking the emails versus X. Thank you for switching." He said: "I do lots of Python programming and really like the code

creation process. So I don't use ChatGPT to write my initial code, but I use it after I've written a function. I just paste it in and ask ChatGPT to describe what it does. If I like the result, I ask it if there is any way to improve the code I've written." He said: "I do have my ChatGPT customized so that it prefers readability, descriptive function/variable names, et cetera, over shorter or potentially more cryptic code. This process fits well into my development flow and results in higher quality code." He said: "I hope this can help other people. It's been working well for me. Alain."

Okay. One of the things coders are always being told is that there's no better way to improve one's craft than to spend serious time reading other people's code. Successful novelists will have always spent their early lives reading other people's novels, and music composers grew up listening intently to endless compositions that preceded them. So it should be no surprise that reading others' code would be every bit as valuable to coders. It's for this reason that I think Alain's idea is very interesting and useful.

ChatGPT has already been trained by reading vast quantities of other people's code. So I think it absolutely makes sense to ask an AI like ChatGPT whether it can see any way to improve upon code that was just written. And that appeals to me far more than asking it to do the work first. If you're coding for the sheer pleasure of doing so, as certainly Alain has said he is, and as I do, then don't give that up. But then also take the opportunity to learn by testing your creation against the distilled wisdom of everyone who previously posted their code to the Internet and influenced ChatGPT's training model. I think that makes a lot of sense.

Leo: Yeah, in fact what I do when I do - it's coming up, by the way, the Advent of Code, December 1st, oh boy.

Steve: Ah.

Leo: Our annual 25-day coding challenge, which I have yet to finish. Came to day 22 last year, so I'm hoping to get to 25 this year. But one of the things I often do is I write - I like to write it first, without looking at anybody's code. But then, you know, I look at all the other people who solved it, and look at ways they solved it, and very often get great ideas, great insights. And if I can find some people doing it in Common Lisp - there are a handful.

Steve: Are there.

Leo: I love looking at how they do it because that really - that's been the best way to improve my Common Lisp is to look at, oh, these masters and the graybeards and the stuff they do. It's very amazing. Not just clever, but really, you know, smart. I love it, yeah. Would you like to take a break before we get to IPv6?

Steve: Let's do it so we don't interrupt this by our last break.

Leo: No. We don't want to interrupt.

Steve: And I think everyone's going to find this really interesting. There are some new thoughts in here that are intriguing.

Leo: Well, I mean, every device I have now pretty much will handle IPv6. And I can use IPv6 addresses and so forth. But there doesn't seem to be the same pressure to give up IPv4 there was for so long.

Steve: Less than half of the top 1,000 websites today can be reached by IPv6.

Leo: Interesting.

Steve: Less than half of the top 1,000.

Leo: Yeah. And we've talked about this before. Vint Cerf and others did not anticipate the success of carrier NAT and ISPs using, you know, NAT at their end. Now, speaking of an idea whose time hasn't come, it keeps coming but hasn't arrived, IPv6, Steve.

Steve: I know that the majority of our listeners need no introduction to the difference between IPv4 and IPv6. But I want to share some of a wonderful recent blog posting made by APNIC Labs. And since it assumes complete comfort with IPv4 vs. IPv6, I want to first share a very quick orientation.

IP stands for Internet Protocol, and Version 4 of the Internet Protocol is the original version that took off and became the worldwide standard. By the mid 1990s, the folks who created this first successful Internet were already starting to worry about its growth because the growth was exponential at that point. So they started working on its successor replacement. That became known as IPv6, or Version 6 of the Internet Protocol.

Although IPv6 changes a bunch of sort of insignificant things from IPv4, the most prominent and significant is addressing. Internet addresses are expressed by a set of binary bits, and any set of binary bits can only have so many possible combinations. The original IPv4 protocol uses 32 bits.

Leo: The original dotted quad. It's four 256, or four eight-bit numbers.

Steve: Four sets of eight bits, exactly.

Leo: Yeah, yeah.

Steve: So back before the Internet happened, when it was still just a "what if" experiment, it was believed that these 32 bits, which allowed for 4,294,967,296 individual Internet addresses...

Leo: We'll never need more than that.

Steve: No, almost 4.3 billion. C'mon, get serious. Right. I mean, what, we had five mainframe computers or something back then.

Leo: Yeah, what they didn't anticipate is that Leo Laporte would have 100 IP-based devices in his house alone.

Steve: Ah, that's true.

Leo: Right?

Steve: So, you know, they thought that would be more than ample. Okay. But as we're going to find out, today around 20 billion devices are attached to the Internet, and many people feel that the Internet is in trouble. If anyone wonders how this is possible, consider the number of Internet-connected devices in the average home, to your point, Leo. And thanks to the miracle of NAT routing - Network Address Translation (NAT) - they're all able to comfortably share the household's single ISP-assigned IP address, in the case of IPv4.

So the way to think about this is that the IPv4 protocol also set aside 16 bits for port numbers. Thus at any given 32-bit IPv4 address, an additional 16 bits are then used to specify the port number at that address. So when you think about this, if you think about the Internet as publicly addressing by port number rather than by host IP, port-based addressing yields an effective 48 bits of total addressing - 32 bits for the IP plus 16 bits for the port at that IP. Thus, what NAT routing does is borrow bits from IPv4's port numbering and reuse them as additional addressing bits. This works, but it really upsets the Internet purists. These guys hate the idea with a passion because, you know, they just say "That's not the way we designed it to work."

Leo: Oh. I didn't know that. Didn't know - they don't like ports, huh?

Steve: They are not happy. In fact, I've got some quotes from them here. They are not happy about that. So, okay. Refocusing on today's topic, everyone agrees that IPv4 is being stretched, and stretched way past its expected end of life. But why? We've had IPv6 since the 1990s. So what's the holdup? At this point, two podcasts away from Episode 1000, would any of our listeners be surprised to learn that it's nothing more than resistance and inertia and the fact that port addressing works well enough?

Okay. So first of all, who are the people who wrote this blog posting? What is APNIC? APNIC is the regional Internet address registry for the Asia-Pacific region, thus AP. It's one of the world's five Regional Internet Registries, abbreviated RIRs. So we can think of this as where the IP address assignments come from because, well, it's where they come from. So here's what the guys in charge of the IP address space have to say as of one week ago, last Tuesday, when this was written.

And since Geoff writes in the first person, it only seems right to introduce him by name as Geoff Huston. He's the Chief Scientist at APNIC, where he undertakes research on Internet infrastructure, IP technologies, and address distribution policies, among other topics. He is widely regarded as the preeminent researcher on IPv4 exhaustion, and is routinely referenced by international agencies and frequently quoted by the media. So Geoff is the guy we want to hear from about this. Here's what he had to say last Tuesday.

He said: "I wrote an article in May 2022, asking 'Are we there yet?' about the transition to IPv6. At the time, I concluded the article on an optimistic note, observing that we may not be ending the transition just yet, but we're closing in. I thought at the time that we wouldn't reach the end of this transition to IPv6 with a bang, but with a whimper. A couple of years later, I'd like to revise these conclusions with some different thoughts about where we are heading and why.

"The state of the transition to IPv6 within the public Internet continues to confound us. RFC 2460, the first complete specification of the IPv6 protocol, was published in December 1998, over 25 years ago. The entire point of IPv6 was to specify a successor protocol to IPv4 due to the prospect of depleting the IPv4 address pool. We depleted the pool of available IPv4 addresses more than a decade ago, yet the Internet is largely sustained through the use of IPv4. The transition to IPv6 has been underway for 25 years, and while the exhaustion of IPv4 addresses should have created a sense of urgency, we've been living with it for so long that we've become desensitized to the issue. It's probably time to ask the question again: How much longer is this transition to IPv6 going to take?

"At APNIC Labs, we've been measuring the uptake of IPv6 for more than a decade now. We use a measurement approach that looks at the network from the perspective of the Internet's user base. What we measure is the proportion of users who can reach a published service when the only means to do so is by using IPv6. The data is gathered using a measurement script embedded in an online ad, and the ad placements are configured to sample a diverse collection of end users on an ongoing basis. The IPv6 adoption report, showing our measurements of IPv6 adoption across the Internet's user base from 2014 to the present, is shown in the Figure." And this is the chart that I have, yup, at the top of this. So it is a very nice-looking, from 2014 to 2020, well, through 2024, so basically a decade, and here we are nearing the end of 2024, so almost 11 years. And it got a little bit of a sluggish start, and then it picked up a little bit in 2017, and then pretty much a straight upward moving line.

Leo: Yeah. That's a good adoption curve. What are the weird spikes, though?

Steve: I think those are just measurement outages.

Leo: Must be errors, yeah, yeah.

Steve: You know, something wasn't working.

Leo: Yeah.

Steve: Okay. So he says: "On the one hand, the figure is one of those classic 'up and to the right' Internet curves that show continual growth in the adoption of IPv6. The problem is in the values in the Y-axis scale. The issue here is that in 2024 we are only at a level where slightly more than one-third of the Internet's user base can access an IPv6-only service. Everyone else is still on an IPv4-only Internet." Only a third are able to access the server.

Leo: That's not good.

Steve: No.

Leo: That's shocking, actually.

Steve: Yeah.

Leo: And those are looking at machines, routers, what are they looking at? What is that?

Steve: So it's a server which is sitting somewhere that only accepts incoming IPv6 traffic.

Leo: So they're looking at receivers versus queriers. Not my machine and my browser. They're looking at the servers themselves.

Steve: Correct. So, and there are, again, you're making a good point. There are many different ways we could consider what does IPv6 adoption mean. So what they're specifically saying is, and he said this here, we're going to chart the percentage of the Internet's user base who are able to reach a service which is only available over IPv6. And right now, as he says, it's one third of users on the Internet can contact a server that you can only get to over v6. And I'll just note that their approach is, I think, very clever. They've scattered ads around the Internet as a means of running a bit of their own script in the user's browser. The script probably queries two servers, one using IPv4 addressing and another using IPv6 addressing. And presumably the visitors whose browsers pull these ads and run this script are widely diverse.

Anyway, Geoff continues. He says: "This seems to be a completely anomalous situation. It's been over a decade since the supply of 'new' IPv4 addresses has been exhausted," meaning there just are no more to give out. "And the Internet," he says, "has not only been running on empty, but also being tasked to span an ever-increasing collection of connected devices without collapsing. In late 2024 it's variously estimated that some 20 billion devices use the Internet, yet the Internet's IPv4 routing table only encompasses some 3.03 billion unique IPv4 addresses."

And I'll just note that the reason for the disparity between the total number of addresses in 32 bits, which is nearly 4.3 billion, and the Internet's current routing table spanning 3.03 billion, is management overhead and the fact that network allocations always leave some headroom. Just, you know, you can have too few hosts in a network. It's not good if you have too many. You can't do that. So here comes the "purist" part of the argument.

Geoff writes: "The original" - and he calls it the "end-to-end" - "the end-to-end architecture of the Internet assumed that every device was uniquely addressed with its own IP address, yet the Internet is now sharing each individual IPv4 address across an average of seven devices, and apparently it all seems to be working. If end-to-end was the sustaining principle of the Internet architecture, then as far as the users of IPv4-based access and services are concerned, it's all over.

"IPv6," he writes, "was meant to address these issues, and the 128-bit wide address fields in the protocol have sufficient address space to allow every connected device to use its own unique address. The design of IPv6 was intentionally very conservative." Meaning

they went way big. They weren't going to make the same mistake twice. He says: "At a basic level, IPv6 is simply 'IPv4 with bigger addresses.' There are also some changes to fragmentation controls, changes to the Address Acquisition Protocols (ARP vs. Neighbor Discovery), and changes to the IP Options fields. But the upper-level transport protocols" - meaning that run on top of IP, the IP packets - "are unchanged. IPv6 was intended to be a largely invisible change to a single level in the protocol stack, and definitely not intended to be a massive shift to an entirely novel networking paradigm.

"In the sense of representing a very modest incremental change to IPv4, IPv6 design achieved its objective. But in so doing it necessarily provided little in the way of any marginal improvement in protocol use and performance. IPv6 was no faster, no more versatile, no more secure than IPv4. The major benefit of IPv6 was to mitigate the future risk of IPv4 pool depletion.

"In most markets, including the Internet, future risks are often heavily discounted." In other words, no one really cares about the future. "The result is that the level of motivation to undertake this transition is highly variable given that the expenditure to deploy this second protocol does not realize tangible benefits in terms of lower cost, greater revenue, or greater market share. In a networking context, where market-based coordination of individual actions is essential, this level of diversity of views on the value of running a dual-stack network leads to reluctance on the part of individual actors and sluggish progress of the common outcome of the transition. As a result, there is no common sense of urgency."

I'll just note that when he refers to a "dual stack," he means using a machine that simultaneously runs both IPv4 and IPv6 protocols, which is entirely possible. Everyone running modern desktop machines today is running a dual stack.

Leo: Yeah. Yeah.

Steve: If I open - what?

Leo: Yeah, I mean, that's how my router is. That's how my desktop is. I can choose IPv6.

Steve: Right.

Leo: I just don't need to.

Steve: Even for me, if I open a command prompt on the Windows 7 machine that's in front of me right now and enter the command "IPConfig," I see that my machine has both IPv4 and IPv6 addresses, as well as IPv4 and IPv6 default gateways. So that means my ISP, Cox Cable, is providing both IPv4 and IPv6 support, which is flowing through my cable modem to my pfSense firewall router, which is distributing both flavors of the Internet to all of the machines in my local network. Thus, dual stack.

So Geoff's point here is that the only significant thing IPv6 was intended to provide, aside from minor fixes around the edges, was significantly greater addressing space. And, inertia being what it is, that was not sufficient to drive its adoption. My guess is what we're seeing is what I would call "adoption by attrition." The same way we're getting

Windows 11 when Windows 10 machines die and it's impossible to get another Windows 10 machine, in other words, for reasons other than desire or demand.

Geoff says: "To illustrate this, we can look at the time series shown in the figure below and ask the question: 'If the growth trend of IPv6 adoption continues at its current rate, how long will it take for every device to be IPv6 capable?'" He says: "This is the same as looking at a linear trend line placed over the data series used in the first Figure, basically extrapolating; right?" He says: "Looking for the date when this trend line reaches 100%. Using a least-squares best fit for this data set from January 2020 to the present day, and using a linear trend line, we come up with Figure 2." And Leo, you've got that on the screen, and it's in the show notes.

"This exercise predicts that we'll see completion of this transition in late 2045, or some 20 years into the future." And I'll just take - I'll take issue with that, but we'll get to that in a minute. I don't think we will ever be there. He says: "It must be noted that there's no deep modeling of the actions of various service providers, consumers, and network entities behind this prediction. The only assumption that drives this prediction is that the forces that shaped the immediate recent past are unaltered when looking into the future. In other words, this exercise simply assumes that 'tomorrow is going to be a lot like today.'

"The projected date in the second Figure is less of a concern than the observation that this model predicts a continuation of this transition for a further two decades. If the goal of IPv6 was to restore a unified address system for all Internet-connected devices, but this model of unique addressing is delayed for 30 years, from around 2015 to 2045, it raises questions about the relevance and value of such a framework in the first place."

Leo: And Steve, I'm going to point out that you and I have some idea of what 20 years means, and it's sooner than you think.

Steve: That is true.

Leo: Right?

Steve: That is true.

Leo: I mean, we are approaching Episode 1000 in two episodes.

Steve: And that will mean that we'll be at 2000 when we...

Leo: In 20 years.

Steve: In 20 years when this model...

Leo: IPv6. And then...

Steve: ...finally says...

Leo: ...we can convert the whole thing to a coloned, what is it, coloned sextet.

Steve: I hate those addresses, Leo.

Leo: They're so ugly.

Steve: They just make your eyes cross.

Leo: They're hex, first of all. They're hex, and they're four hex digits separated by colons. And then, what, are those six groups?

Steve: Well, and they're so long that there's weird, like, abbreviation systems have been created in order to collapse them.

Leo: Oh, I know, I hate the abbreviations. Because there's a lot of zeroes in many IPv6 addresses, so you just collapse those, yeah.

Steve: Yeah. It's not good.

Leo: Not good.

Steve: So he says: "If we can operate a fully functional Internet without such a coherent end-device address architecture for three decades, then why would we feel the need to restore address coherence at some future point in the future? What's the point of IPv6 if it's not address coherence? Something," he writes, "has gone very wrong with this IPv6 transition, and that's what I'd like to examine in this article."

Okay. So he goes on at great length, more than this podcast even can handle. So I'm going to skip some things, but I'm going to share some highlights. Let's look back a bit to see what these Internet pioneers saw during the 1990s. He says: "By 1990 it was clear that IP had a problem. It was still a tiny Internet at the time, but the growth patterns were exponential, doubling in size every 12 months." Now, there are two things that have happened that they did not foresee. And those two things solved this problem. NAT's only one of them. NAT's only on the client side.

He says: "We were stressing out the pool of Class B IPv4 addresses, and in the absence of any corrective measures this address pool would be fully depleted in 1994." Okay. So they were at 1990. And they were charting the rate of Class B network allocation consumption. And I have a picture here that was taken, it was from the proceedings of the IETF in August 1990. And it's so quaint because they were still, like, drawing things by hand. Yu know, it's like written out, you know, by hand. Just, you know, adorable. Wow.

Leo: Back in 1990 we really didn't have laser printers. We had to do it by hand. It's like a back of a napkin.

Steve: Yeah. And those are from the official proceedings. It's titled "Internet Growth" by Frank...

Leo: Solensky.

Steve: ...Solensky, proceedings of the IETF August 1990.

Leo: At least Frank used a ruler for the graph.

Steve: Yeah, he did, yeah. But not the title and the headline.

Leo: No.

Steve: You know, and other notations.

Leo: No, have to rewind it by hand.

Steve: So Geoff explains that the IETF was panicking in the early 1990s because the Internet's original design was designed, it was destined to collapse. Remember, Leo, back then there were only three classes of network allocation. And that was a big problem.

He says: "There was a collection of short, medium, and long-term responses that were adopted in the IETF to address the problem. In the short term, the IETF dispensed with the class-based IPv4 address plan and instead adopted a variably sized address prefix model." He said: "Routing protocols, including BGP, were quickly modified to support these classless address prefixes. Variably sized address prefixes added additional burdens to the address allocation process, and in the medium term, the Internet community adopted the organizational measure of the Regional Internet Registry structure to allow each region to resource the increasingly detailed operation of address allocation and registry functions for their region.

"These measures increased the specificity of address allocations and provided the allocation process with a more exact alignment to determine adequate resource allocations that permitted a more diligent application of relatively conservative address allocation practices. These measures realized a significant increase in address utilization efficiency. The concept of 'address sharing' using Network Address Translation (NATs) also gained some traction in the ISP world. Not only did this dramatically simplify the address administration processes in ISPs, but NATs also played a major role in reducing the pressures on overall address consumption.

"The adoption of these measures across the early 1990s pushed a two-year imminent crisis into a more manageable decade-long scenario of depletion. However, they were not considered to be a stable long-term response. It was thought at the time that an effective long-term response really needed to extend the 32-bit address field used in IPv4. At the time, the transition from mainframe to laptop" - from mainframe, Leo. Mainframes were on the Internet.

Leo: There were only a dozen of them.

Steve: That's right.

Leo: They had a couple of hundred amps, and that was it.

Steve: "So the transition from mainframe to laptop was well underway in the computing world, and the prospect of further reductions in size and expansion of deployment in smaller embedded devices was clear at the time. An address space of four billion was just not large enough for what was likely to occur in the coming years in the computing world." But of course if you absolutely did require every device to have their own address...

Leo: That's what - yeah.

Steve: Absolutely true. We are at 20 billion and growing fast today.

Leo: Easy, yeah. Yeah. Well, you know, I mean, I can imagine somebody saying, oh, my god, they're giving toasters their own Internet address. We've got a problem here. It's going to be awful.

Steve: It's going to die.

Leo: I mean, laptops, forget laptops. What about IoT? I mean...

Steve: Yes.

Leo: This is about to explode. You have light switches that have IP addresses.

Steve: Yes, exactly. So to the point he just made about class A, B, and C networks, we should remember that the original Internet divided the entire network space, 32 bits, on byte boundaries. IPv4 addresses have, as we said, four 8-bit bytes. So a class A network was numbered by its most significant byte. The most significant byte was the network number, so you couldn't have many of them. And then the remaining 24 bits to the right of that most significant byte were the host machine within that massive network.

Leo: So you could only have 255...

Steve: Class A networks.

Leo: ...or 256 class A networks, yeah.

Steve: Right.

Leo: Total.

Steve: Class B networks used two bytes for the network ID and then 16 bits for the individual host machines within each one of those class B networks. And finally class C networks had three bytes for their network ID and then just one byte for host machines. So they could only have 254 because you need all zeroes and all ones are reserved for broadcast and things.

So anyway, the problem that Geoff is referring to is that this created massive granularity, massively granular network allocations. The adoption of the so-called "Classless," because you don't have classes A, B, and C, "Classless Inter-Domain Routing," or CIDR, where the division between the network ID on the left and the host machine's number in that network on the right could now fall on any bit boundary, rather than being only on byte boundaries. That massively increased the load on the Internet's routers and on the routing tables. But in return, it meant that the size of individual network allocation could much more closely track and better fit the number of host machines within that network. So that was a huge win that bought them a decade, basically, because, I mean, otherwise they would, you know, just couldn't have that many networks, let alone that many machines.

But Geoff mentioned the emergence of NAT routing, and a fascination of mine has always been "what's wrong with NAT?" It works.

Leo: Yeah. We're all using it.

Steve: Oh, my god, we have to have it.

Leo: Yeah.

Steve: Here's what Geoff has to say about NAT. He says: "At this point, there was no choice for the Internet, and to sustain growth in the IPv4 network while we were waiting for IPv6 to gather momentum, we turned to NATs. NATs were a challenging subject for the IETF. The entire concept of coherent end-to-end communications was to eschew active middleware in the network."

Leo: They wanted everything to have a unique address, every single thing on the network.

Steve: The original concept was point to point, address to address. And they did not want to let that go.

Leo: It's be like having phone numbers without area codes. It would just be, you know, limited.

Steve: Yup. They just, they said, this is wrong. This is not the way it's supposed to be.

Leo: Interesting.

Steve: So he says: "NATs created a point of disruption in this model, creating a critical dependency upon network elements. They removed elements of network flexibility from the network and at the same time reduced the set of transport options to TCP and UDP."

Leo: Huh.

Steve: And when you think about it, you can't ping, like, arbitrary devices behind a NAT router.

Leo: Right, they're hidden.

Steve: And you're supposed to be able to ping any device on the Internet.

Leo: It really makes you think. If they had adopted IPv6 from the very first, it would be a very different network.

Steve: Oh, it would be completely different, yes.

Leo: So many other things would be possible.

Steve: Yes.

Leo: Yeah.

Steve: Many, many other things. That's exactly right.

Leo: You could finger every device, as it were. But you could query devices. Very interesting.

Steve: It would all be publicly available.

Leo: Security would be maybe a little more challenging. I mean, that protects us, doesn't it, behind the firewall.

Steve: Oh, my god, yes. It is a wonderful firewall technology.

Leo: Yeah.

Steve: But the fact that it's a firewall, like as a side effect, is their complaint.

Leo: Yeah, they didn't like that.

Steve: You could have one, but it shouldn't be like there's no...

Leo: You have to.

Steve: You cannot not have one, exactly.

Leo: Right, right.

Steve: Yeah. And as we know, you cannot put a machine on the raw Internet today.

Leo: Oh, my god.

Steve: It's taken over in seconds.

Leo: Yeah.

Steve: Okay. So he says: "The IETF resisted any efforts to standardize the behavior of NATs, fearing perhaps that standard specifications of NAT behavior would bestow legitimacy on the use of NATs, an outcome that several IETF participants" - and you know they have beards - "were very keen to avoid." He said: "This aversion did not reduce the level of impetus behind NAT development." In other words, sorry, we don't care what you guys don't like.

Leo: We have to.

Steve: We need them.

Leo: Yes.

Steve: He said: "We had run out of IPv4 addresses, and IPv6 was still a distant prospect, so NATs were the most convenient solution. What this action did achieve was to create a large variance of NAT behaviors in various implementations. "In other words, since they were unwilling to standardize them, what we just got was a mess because everyone just had to invent this stuff for themselves, and everybody did it a little bit differently. He said: "What this action did achieve was to create a large variance of NAT behaviors in various implementations, particularly concerning UDP behaviors. This has exacted a cost in software complexity where an application needs to dynamically discover the type of NAT or NATs in the network path if it wants to perform anything more complex than a simple two-party TCP connection.

"Despite these issues, NATs were a low-friction response to IPv4 address depletion where individual deployment could be undertaken without incurring external dependencies. On the other hand, the deployment of IPv6 was dependent on other networks and servers also deploying IPv6. NATs made highly efficient use of address space for clients, as not only could a NAT use the 16-bit source port field, but by time-sharing the NAT binding, NATs achieved an even greater level of address efficiency, basically reusing the space. A major reason why we've been able to sustain an Internet with tens of billions of connected devices is the widespread use of NATs."

Okay. So that's over on the client side of connections. The solutions that the industry has evolved over on the server side is something we've covered previously, but never really thought about in this context. Geoff writes: "Server architectures were evolving, as well. With the introduction of TLS (Transport Layer Security) in web servers, a step was added during TLS session establishment where the client informs the server of the service name it intends to connect to. Not only did this allow TLS to validate the authenticity of the service point, but this also allowed a server platform to host an extremely large collection of services from a single platform and a single platform IP address, and perform individual service selection via this TLS Server Name Indication (SNI).

"The result is that server platforms perform service selection by name-based distinguishers (DNS names) in the session handshake, allowing a single server platform to serve large numbers of individual servers. The implications of the widespread use of NATs for clients and the use of server sharing in service platforms have taken the pressure off the entire IPv4 address environment."

And I have a perfect example of this at GRC. I don't have endless IPs given to me from Level 3. You know, I'm clutching the set that I have dearly. But through the years, the range of services I have wanted to offer has grown. Thanks to server name indication, I have, I just checked, 13 different web services sharing a single IP address. DNS points 13 different domains to a single IP. And any web browser that wishes to connect indicates the machine it's looking for during that connection handshake.

So that's really something I hadn't focused on, but it is absolutely true. Both ends of the IPv4 connection. The client-side has NAT that allows, for practical purposes, limitless expansion there on the client-side. And on the server side, SNI allows hosting providers to have a modest number of IP addresses. DNS is now redundant. It's redundantly pointing a huge array of DNS names at a subset, at a small number of IPv4 addresses. And all of this disambiguation from domain name to IP address occurs thanks to the TLS-SNI handshake where the browser says this is the host I'm looking for. I'm told it's at this IP address. Well, yes. It and hundreds of others are all there. So it's a really cool scheme, and it actually works.

Okay. So Geoff goes on in substantially greater detail for anyone who's interested. In the interests of time, as I said, I've deliberately skipped over a lot of Geoff's truly interesting discussion. But he eventually gets to examining the question: "How much longer?" He says: "Now that we are somewhere in the middle of this transition, the question becomes, 'How much longer is this transition going to take?'"

He says: "This seems like a simple question, but it does need a little more explanation. What is the 'endpoint' when we can declare this transition to be complete? Is it a time when there is no more IPv4-based traffic on the Internet? Is it a time when there is no requirement for IPv4 in public services on the Internet? Or do we mean the point when IPv6-only services are viable? Or perhaps we should look at the market for IPv4 addresses and define the endpoint of this transition at the time when the price of acquiring a new IPv4 address completely collapses?

"Perhaps we should take a more pragmatic approach and, instead of defining completion as the total elimination of IPv4, we could consider it complete when IPv4 is no longer necessary. This would imply that when a service provider can operate a viable Internet service using only IPv6 and having no supported IPv4 access mechanisms at all, then we would've completed this transition.

"What does this imply? Certainly, the ISP needs to provide IPv6." Obviously. "But, as well, all the connected edge networks and the hosts in these networks also need to support IPv6. After all, the ISP has no IPv4 services at this point of completion of the transition. It also implies that all the services used by the clients of this ISP must be accessible over IPv6. Yes, this includes all the popular cloud services and cloud platforms, all the content streamers, and all the content distribution platforms. It also includes specialized platforms such as Slack, Xero, Atlassian, and similar.

"The data published on Internet Society's Pulse reports that only 47% of the top 1,000 websites are reachable over IPv6 today, and clearly a lot of service platforms have work to do. And this will take more time. When we look at the IPv6 adoption data for the U.S., there's another curious anomaly." And Leo, that's the last chart that I talked about. Look at that. I think it's very interesting.

Leo: It's flat.

Steve: It is flat since a little bit into 2019. It's stopped growing.

Leo: Oh, boy.

Steve: It went, in 2014, it came off the ground at about a little over, looks like a little over 5%, maybe 6%; climbed up to around 55-60; and then flat-line. I know that we've previously...

Leo: So this is websites.

Steve: That, no. This is their probe which showed linear growth. Same probe shows for U.S. it is flat.

Leo: Oh, this is U.S. compared to the global graph that we saw previously. Ah.

Steve: Correct. Correct. I know that we've previously observed that much of the IPv6 growth has been elsewhere in the world.

Leo: Yeah, not around here.

Steve: Developing nations, for example, which are just obtaining Internet access, are naturally acquiring IPv6 access since they have no inertia, and IPv6 is certainly available. But where we previously observed a surprisingly straight upward moving line of total global adoption, the chart showing only U.S.-based adoption is an entirely different

animal. For the past six years, since around the start of 2019 and through 2024, the United States IPv6 has been flat, showing no growth. None.

Geoff draws the really interesting conclusion that the services and the service model of the Internet are changing; and that, in a very real sense, DNS has evolved into our routing protocol, alluding to what I mentioned before. He explains.

He says: "It's domain names that operate as service identifiers." It was supposed to be IPs. No. It's domain names that operate as service identifiers. And this is him: "And it's domain names that underpin the user tests of authenticity of the online service. It's the DNS that increasingly is used to steer users to the best service delivery point for content or service. From this perspective, addresses IPv4 or IPv6, are not the critical resource for a service and its users. The currency of this form of CDN network is names.

"So where are we in 2024? Today's public Internet is largely a service delivery network using CDNs to push content and service as close to the user edge as possible. The multiplexing of multiple services onto underlying service platforms is an application-level function tied largely to TLS and service selection using the SNI field of the TLS handshake. We use DNS for 'closest match' service platform selection, aiming for CDNs to connect directly to the access networks where users are located. This results in a CDN's routing table with an average path length designed to converge on one. From this aspect, the DNS has supplanted the role of routing. While we don't route names on today's Internet, it functions in a way that is largely equivalent to a named data network. In other words, no longer addresses, but names.

"There are a few additional implications of this architectural change for the Internet. TLS, like it or not, and there is much to criticize about the robustness of TLS, is the sole underpinning of the authenticity in the Internet. DNSSEC has not gathered much momentum to date. DNSSEC is too complex, too fragile, and just too slow to use for most services and their users. Some value its benefits highly enough that they are prepared to live with its shortcomings, but that's not the case for most name holders and most users, and no amount of passionate exhortations about DNSSEC will change this. It supports the view that it's not the mapping of a name to an IP address that's critical. What is critical is that the named service can demonstrate that it is operated by the owner of the name." In other words, certificates.

"Secondly, the Routing PKI, the framework for securing information being passed in the BGP routing protocol, is really not all that useful in a network where there is no routing. The implication of these observations is that the transition to IPv6 is progressing very slowly, not because this industry is chronically short-sighted. There is something else going on here. IPv6 alone is not critical to a large set of end-user service delivery environments. We've been able to take a 1980s address-based architecture and scale it more than a billion-fold by altering the core reliance on distinguisher tokens from addresses to names. There was no real lasting benefit in trying to leap across to just another 1980s address-based architecture, meaning IPv6, with only a few annoying stupid differences, apart from longer addresses."

So to give this something of a summary, what's happened is that the Internet has become the WebNet. It is mostly all about the World Wide Web. And even where it isn't, most endpoints are still being secured by the web's TLS. What's happened is that both ends of the web have independently solved their IPv4 resource depletion problem. Over on the client end we have NAT routing which, as we've noted earlier, effectively borrows excess bits from the 16 bits of port addressing to allow many client-side devices to share a single public 32-bit IPv4 address. And over on the server side we have Server Name Indication (SNI) which allows GRC, for example, to host 13 different named services from a single IP address. Name is the key. That the key, and this is the point that I think Geoff

brilliantly observes, we are now using names rather than addresses to access the services we need.

And to see that, you know, on top of that, fewer than half of the top 1,000 websites are reachable at all over IPv6. Certainly all of them over IPv4. But IPv6, fewer than half. That suggests that the majority still feel very little pressure to invest in something that will literally make no difference in the services they deliver.

And finally, even before seeing that the U.S. adoption of IPv6 has been completely flat and static for the past five years, we know that no straight line continues straight out to the end. That's just not the way the world works. That line was a percentage of IPv6 adoption, so that rate of adoption is absolutely going to slow down, and probably not long from now. Nothing gets to 100%. So my guess is that it will begin flattening out and will asymptotically approach 90% over a great many more decades. And that's fine, too, since I think it's clear that IPv4 will never die.

Leo: This should be the title of the show. IPv4 will never die. Well, that's, you know what, you were right. I was thinking, is Steve really going to be able to turn this into something interesting? And it is, it's quite interesting, actually. And the way that the Internet has routed around the problem and solved it kind of organically and effectively is very, very interesting. So the real issue is without the pressure to go to IPv6, nobody's - it's like metric. It's no accident that the U.S. is a laggard here.

Steve: No. And notice that it's in our desktops. We didn't ask for it. But it's just there. So...

Leo: Is it easy to build it in? I mean, is it the kind of thing where it's, well, we can implement it on the client end easily?

Steve: Yeah. I mean, there's open source code, all of the various, you know, IP stacks support it now. So it's just there. And so it'll end up getting used. There is a preferential use of it when both are available. IPv6 is chosen so that 4 is now become the fallback.

Leo: I had heard, it's probably apocryphal, that IPv6 is faster.

Steve: No, it's not faster.

Leo: It's the same.

Steve: There's nothing about it. You could argue it's a little slower because it's got a little more addressing overhead.

Leo: Right, yeah.

Steve: And that's his point. If it was faster, it would have been adopted.

Leo: Right, there just was no real pressure to do it.

Steve: All it is is offering something that it turns out we don't need.

Leo: So do you think we'll never get there?

Steve: Well, I'm still nervous about the client end because four billion, you know, we still need four billion clients. Now, carrier NAT, as you said, carrier NAT solves that. Right now I have a public IP address, you know, Cox is like 38 dot something dot something, or 70 dot something something. So I have a public IPv4 address. Some people are getting 10-dot addresses from their ISPs.

Leo: They're NATing them.

Steve: The ISP is doing the NAT. And so it's double NAT. ISP is NATed, and they get one IP, and then their residential NAT router is NATing. But so again, it solves the problem. If the ISP has more customers than it's able to get IPs from its upstream supplier, it just applies carrier-grade NAT.

Leo: Fascinating.